BLOCK-DECOMPOSITION AND ACCELERATED GRADIENT METHODS FOR LARGE-SCALE CONVEX OPTIMIZATION

A Thesis
Presented to
The Academic Faculty

by

Camilo Ortiz

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
May 2014

BLOCK-DECOMPOSITION AND ACCELERATED GRADIENT METHODS FOR LARGE-SCALE CONVEX OPTIMIZATION

Approved by:

Professor Renato D. C. Monteiro,
Advisor
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Arkadi Nemirovski
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Eva K. Lee
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Benar F. Svaiter
IMPA
Instituto Nacional de Matemática Pura e Aplicada

Professor Shabbir Ahmed
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Date Approved: March 10, 2014

To Tati.

ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor, Prof. Renato D. C. Monteiro, for his close involvement and mentorship throughout my Ph.D. program. The support and guidance received from Prof. Monteiro have been fundamental for the completion of this thesis. His thoroughness and keen eye for research were an inspiration to keep pushing the limits on our results. Most of all, I am grateful for the countless hours that he spent in our collaborations, coaching me through all the processes involved in performing robust research.

My research was also extremely benefitted by our productive collaborations with Prof. Benar F. Svaiter. His fast-passed research and quick inventiveness were really exemplary every time we worked together.

I would like to express my gratitude to all the professors that taught and worked with me at Georgia Tech. I thank Professors Eva K. Lee, Shabbir Ahmed and Arkadi Nemirovski for serving as members of my thesis committee. Also, I am grateful to Professors Ahmed and Nemirovski that, together with Professors Craig Tovey and George Nemhauser, provided me with the most complete and enjoyable set of optimization classes that I could ever hope for.

My experience in Atlanta has been specially wonderful because of the great friendships forged along the way. I was lucky to start the Ph.D. program with my (now) great friends Vinod Cheriyan, Wonki Kim, Todd Levin and Monica Villareal. I was also fortunate to get to know friends and fellow graduate students Brian Kues, Wajdi Tekaya, Dimitri Papageorgiou, Fatma Kılınç-Karzan, George Thiers, Pratik Mital, Min Kyu Sim, Jeff Pavelka, Rodrigue Ngueyep Tzoumpe, Mallory Soldner, Daniel Silva, Diego Morán, Tim Sprock and many others whom I shouldn't have forgotten. Additionally, I am extremely grateful and in debt for the camaraderie and support to all my other "judas" friends: Carlos Campo, whose friendship I have greatly appreciated since my first year, Felipe Castrillón, Ciro Espinosa, David Gonzalez, Barbara Piña, Maria Restrepo, Giovanny Rozo, Vivian Soler and

Alejandro Suarez.

I would like to thank my parents, Ricardo and Blanca, and my brother, David, for their love, huge support and advise. David has constantly been my example to always try to enjoy every life experience. Also, I want to thank my hometown family and friends, specially Sandra, Pepina and Edgar, whom I have always been able to count on when I need.

I have no words that can reflect how fortunate I am to have had next to me, in every part of this experience, the most important person of my life, my best friend and future wife, Tatiana. Thank you for all of these fantastic years where we managed to successfully create our own home.

Finally, I would like to acknowledge the overwhelming support received from the US Department of State with the Fulbright Science and Technology Award. Also, the partial support for this research from the ARC Fellowship from Georgia Tech.

TABLE OF CONTENTS

List of Tables

List of Figures

SUMMARY

In recent years, there has been an increased interest in the development of new and improved first-order optimization methods for minimizing smooth and non-smooth convex functions. Every year, multiple applications are found which require specialized methods that solve large non-linear convex optimization problems. These include applications in statistics, machine learning, imaging and signal processing, and the solution of good conic relaxations of various NP-hard combinatorial problems including: max-clique problems; frequency assignment problems (or max k-cut problem); quadratic assignment problems; traveling salesman problems; and general binary integer quadratic problems. Even though significant progress has been made in sophisticated interior point (second-order) methods (e.g., several variants and combinations of a Newton-type method with a technique to solve linear equations), when it comes to solving large-scale problems, involving millions of variables and constraints, these methods fail to even perform one iteration using reasonable amounts of resources (time and memory). On the other hand, the development of efficient first-order methods has been fundamental in satisfying the increasing demand for handling larger instances. What first-order methods lack in fast convergence properties like the ones for IPMs, they compensate with iterations that perform cheap operations (function values and gradients).

In 1983, Y. Nesterov presented a scheme for accelerating first-order methods, showing that the rate of convergence for these methods, namely $\mathcal{O}(1/k)$, can be improved to $\mathcal{O}(1/k^2)$ by considering their accelerated variants, where k denotes the iteration count. Due to its wide use for solving large-scale convex optimization problems, several other accelerated variants have been proposed and studied in the literature. Moreover, there has been an increasing concern of improving the practical performance of these accelerated methods to find solutions to large-scale problems. In fact, if the problem has a special structure,

e.g., the problem is conic, the current most efficient methods/codes in practice are first-order projection-type (proximal) methods. These latter methods are not accelerated and, in general, have a theoretical rate of convergence $\mathcal{O}(1/k)$. The idea behind these methods is to split the problem with a block-decomposition (BD) or splitting operator approach in order to solve easy subproblems at every iteration.

In the first part of this thesis, we consider *exact BD methods* for conic semidefinite programming (SDP). More specifically, we propose two proximal point BD methods that *exactly* solve both of the proximal subproblems corresponding to a two-block reformulation of the optimality conditions. Both of these BD methods are applications of a BD framework that is in the context block-structure monotone inclusion problems which in turn is based on the hybrid proximal extragradient (HPE) method introduced in 1999 by Solodov and Svaiter, and whose complexity is derived in 2010 by Monteiro and Svaiter. The first exact BD method is derived from a two-block reformulation of the optimality conditions of standard form conic SDP problems. The second one is derived from a two-block reformulation of the optimality conditions of convex optimization problems that consist of minimizing the sum of a convex differentiable function with Lipschitz continuous gradient, and two other proper closed convex (possibly, nonsmooth) functions. In addition to being able to solve standard form conic SDP problems, the latter method is also able to directly solve specially structured non-standard form conic programming problems without the need to add additional variables and/or constraints to bring them into standard form. The BD framework and the exact BD methods are presented with several new speed-up refinements from a computational point of view that include adaptive (aggressive) choices of extragradient stepsizes and the use of a scaling factors to balance the blocks. Finally, we present computational results on several classes of SDPs showing that our exact BD methods outperform the three most competitive codes for (standard and non-standard form) large-scale conic semidefinite programming.

In the second part of this thesis, we consider an *inexact BD method* for solving extra large-scale conic SDP problems. Similar to the above exact BD methods, this inexact BD method is an instance of a proximal point BD framework that is based on the HPE

method. However, in contrast to the exact BD methods, this BD method *inexactly* solves the proximal subproblem corresponding to the second block which, in this case, consists of both the primal and dual affine feasibility constraints. As a result, the inexact BD method avoids computations of exact projections onto the manifold defined by the affine constraints. Moreover, we propose a scheme that uses the conjugate gradient (CG) method applied to a reduced positive definite dual linear system to obtain inexact solutions of the augmented primal-dual linear system corresponding to the second block. The proposed BD method implements the latter scheme and also incorporates a new dynamic scaling scheme that uses two scaling factors to balance three inclusions comprising the optimality conditions of the conic SDP. Finally, we present computational results showing the efficiency of our inexact BD method for solving various extra large SDP instances, several of which cannot be solved by other existing methods, including some with at least two million constraints and/or fifty million non-zero coefficients in the affine constraints.

In the third part of this thesis, we consider an adaptive accelerated gradient method for a general class of convex optimization problems. First, we define a general accelerated framework for convex optimization which (basically) outlines sufficient conditions to obtain $\mathcal{O}(1/k^2)$ convergence on the functional gap for constant stepsizes. Then, we introduce a scheme that adaptively and aggressively chooses certain acceleration parameters of the framework in order to substantially improve its practical performance without compromising its theoretical iteration-complexity. The proposed adaptive accelerated method, which can be viewed as a new variant of Nesterov's method, is a simple application of the accelerated framework that incorporates the latter scheme together with an adaptive restart scheme (on the acceleration parameters). Finally, we present numerical results demonstrating that the proposed adaptive accelerated method performs quite well compared to other variants proposed earlier in the literature.

Chapter I

INTRODUCTION

In this chapter, we describe the main problems of interest and review some general background and recent developments on large-scale optimization. This chapter is divided into four sections. Section 1.1 reviews the advancements on methods for large-scale conic programming, including block-decomposition (BD) methods which are the main focus of our research, and gives a general background on inexact proximal point methods. Section 1.2 describes the general convex optimization problem, and reviews the background and recent advancements on accelerated gradient methods. Section 1.3 describes the organization of this thesis and Section 1.4 introduces some common notation.

1.1 Large-scale conic programming

Let $\mathcal{X}$ and $\mathcal{Y}$ be finite dimensional inner product spaces, with inner products and associated norms denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively. We are interested in the conic programming problem

$$\min\{\langle c, x\rangle \ : \ \mathcal{A}x = b, \ x \in K\}, \tag{1.1.1}$$

where $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is a linear mapping, $c \in \mathcal{X}$, $b \in \mathcal{Y}$, and $K \subseteq \mathcal{X}$ is a closed convex cone. The corresponding dual problem is

$$\max\{\langle b, y\rangle \ : \ c - \mathcal{A}^* y \in K^*\}, \tag{1.1.2}$$

where $\mathcal{A}^*$ denotes the adjoint of $\mathcal{A}$ and K^* is the dual cone of K defined as

$$K^* := \{v \in \mathcal{X} : \langle x, v\rangle \geq 0, \ \forall x \in K\}. \tag{1.1.3}$$

Recent research on conic programming has particularly focused on the case when K is the direct product of three types of convex cones: the nonnegative orthant $\mathbb{R}^n_+$, the second-order cone $\{x \in \mathbb{R}^{n+1} : \sum_{i=1}^n x_i^2 \leq x_0^2\}$, and the positive semidefinite matrix cone $\mathcal{S}^n_+$. Due

to their extremely wide area of applications, cone programs associated with these cones are sufficiently general to serve as the basis of conic programming modeling packages. Moreover, it is well-known that the positive semidefinite matrix cones are in fact as general as any cone consisting of arbitrary combinations of the above three cone types. For the purpose of this thesis, we will consider the conic semidefinite programming (SDP) problems, i.e., special cases of (1.1.1) in which

$$\mathcal{X} = \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}, \qquad \mathcal{Y} = \mathbb{R}^m, \qquad K = \mathbb{R}^{n_u} \times \mathbb{R}^{n_l}_+ \times \mathcal{S}^{n_s}_+, \tag{1.1.4}$$

and the inner products in $\mathcal{X}$ and $\mathcal{Y}$ are the standard Euclidean/Frobenius inner products.

It is widely accepted that interior-point methods (IPMs) with direct solvers are generally the most efficient and robust for small and medium (m up to a few thousands) sized conic SDP problems (1.1.1) and (1.1.4). These IPMs are Newton-type (second-order) methods that use direct solvers on the system of linear equations that generates new primal (or primal-dual) search directions at each iteration. The limitations of IPMs with direct solvers become very severe due to the need for computing, storing, and factorizing the $m \times m$ Schur complement matrix which is very dense for most applications. Nevertheless, the recent literature contains many examples of slightly larger (m up to a few tens of thousands) conic SDPs that were solved successfully by scalable customized implementations of IPMs (see for example [2, 58, 17, 24, 23, 22, 27, 66]). These results were obtained by a variety of direct and iterative linear algebra techniques that take advantage of non-sparse problem structure. However, since m could be $\mathcal{O}(n_s^2)$, the limitations on the size of m restricts the use of IPMs significantly.

The following subsection reviews alternatives methods to IPMs for large-scale (i.e., $m \geq 10,000$ and/or $n_s \geq 1000$) conic SDPs (1.1.1) and (1.1.4) .

1.1.1 Previous methods for large-scale conic SDPs

The use of nonlinear optimization techniques has been a successful approach to solve large instances of some problem classes of (1.1.1). For instance, the methods in [21, 7, 40, 10] solve (1.1.1) after rewriting them as nonlinear (possibly, nonconvex) programs. Strong

computational results on large problems with medium accuracy have been reported for these algorithms.

In [12, 28, 48], the first-order methods and smoothing techniques of Nesterov [42, 45] and Nemirovski [41] have been applied to certain SDPs with some special structures. More recently, these methods, including new variants of Nesterov's optimal method, were applied to primal-dual reformulations of the general conic programming problem (1.1.1) in [25].

Presently, the most efficient methods/codes for solving large-scale conic SDP problems are the first-order projection-type discussed in [30, 67, 68] (see also [52] for a slight variant of [30]). More specifically, augmented Lagrangian approaches have been proposed for the dual formulation of (1.1.1) with $\mathcal{X}$, $\mathcal{Y}$ and K as in (1.1.4) for the case when m, n_u and n_l are large (up to a few millions) and n_s is moderate (up to a few thousands). In [30, 52], a boundary point (BP) method for solving (1.1.1) is proposed which can be viewed as variants of the alternating direction method of multipliers introduced in [16, 18] applied to the dual formulation (1.1.2). In [68], an inexact augmented Lagrangian method is proposed which solves a reformulation of the augmented Lagrangian subproblem via a semismooth Newton approach combined with the conjugate gradient method. In [67], an efficient variant of the BP method is discussed and numerical results are presented showing its impressive ability to solve important classes of large-scale graph-related SDP problems.

In this thesis, we study alternatives to all the above methods/approaches, using inexact proximal point methods to find approximate solutions for the optimality conditions of (1.1.1).

1.1.2 Proximal point methods and the BD approach

This subsection reviews the exact proximal point method and some of its inexact variants that have been proposed in the literature to solve the monotone inclusion problem.

A broad class of optimization, saddle point, equilibrium, and variational inequality problems can be posed as the monotone inclusion (MI) problem, namely, finding x such that

$$0 \in T(x),$$

where T is a maximal monotone point-to-set operator (see Subsection 2.1.1). The *exact* proximal point method, proposed by Martinet [31], and further generalized by Rockafellar [56, 57], is a classical iterative method for solving the MI problem which, given a sequence of stepsizes $\{\lambda_k > 0\}$, generates a sequence $\{x^k\}$ according to

$$x^k = (\lambda_k T + I)^{-1}(x^{k-1}),$$

or equivalently,

$$v^k \in T(x^k), \qquad \lambda_k v^k + x^k - x^{k-1} = 0. \tag{1.1.5}$$

This approach has been used as a framework for the design and analysis of several implementable algorithms. The classical *inexact* version of the proximal point method allows for the presence of a sequence of summable errors in the above iteration, that is:

$$\|x^k - (\lambda_k T + I)^{-1}(x^{k-1})\| \le e_k, \qquad \sum_{k=1}^{\infty} e_k < \infty.$$

Convergence results under the above error condition have been established in [57] and have been used in the convergence analysis of other methods that can be reformulated as a MI problem.

New inexact versions of the proximal point method with relative error tolerance were proposed by Solodov and Svaiter [59, 60, 61, 62]. Iteration-complexity results for one of these inexact versions of the proximal point method introduced in [59], namely the hybrid proximal extragradient (HPE) method, were established by Monteiro and Svaiter [36]. In each step of the HPE method, the above proximal system (1.1.5) is solved inexactly as follows. For a given constant $\sigma \in [0, 1]$, a stepsize $\lambda = \lambda_k$ and a triple $(\tilde{x}, \tilde{v}, \varepsilon) = (\tilde{x}^k, \tilde{v}^k, \varepsilon_k)$ are found such that

$$\varepsilon \ge 0, \qquad \tilde{v} \in T^{\varepsilon}(\tilde{x}), \qquad \|\lambda\tilde{v} + \tilde{x} - x\|^2 + 2\lambda\varepsilon \le \sigma^2\|\tilde{x} - x\|^2, \tag{1.1.6}$$

where $x = x^{k-1}$ and T^{ε} denotes the ε-enlargement [4] of T. (It has the property that $T^{\varepsilon}(x) \supset T(x)$ for each x; see Subsection 2.1.1 for details.) Note that this construction relaxes both the inclusion and the equation in (1.1.5). In other words, any pair (x^k, v^k) satisfying the exact proximal condition (1.1.5), also satisfies the HPE error condition (1.1.6)

for any $\sigma, \varepsilon \geq 0$. Finally, instead of choosing $\tilde{x}^k$ as the next iterate x^k, the HPE method computes the next iterate $x^+ = x^k$ by means of the following extragradient step:

$$x^+ = x - \lambda \tilde{v}. \tag{1.1.7}$$

Application of this framework to the iteration-complexity analysis of several zero-order (or, in the context of optimization, first-order) methods for solving (generalized) monotone variational inequalities and saddle-point problems were discussed in [36] and in the subsequent papers [37, 39].

In particular, paper [39] derives the iteration-complexity of a block-decomposition HPE (BD-HPE) framework for the monotone inclusion problem consisting of the sum of a continuous monotone map and a point-to-set maximal monotone operator with a separable two-block structure, namely,

$$0 \in T(z,w) := \left\{ \begin{pmatrix} F_1(z,w) + c \\ F_2(z,w) + d \end{pmatrix} : c \in C(z), d \in D(w) \right\}, \tag{1.1.8}$$

by using the fact that it is a special case of the HPE method. The BD-HPE framework allows for each one of the single-block proximal subproblems to be solved in an exact or approximate sense. More specifically, given a pair $((z,w),\lambda)$, an iteration of an *exact* instance of the BD-HPE framework computes an approximate solution $((\tilde{z},\tilde{w}),(\tilde{v}_1,\tilde{v}_2))$ of (1.1.5) with T given by (1.1.8) by first computing an exact solution $(\tilde{z},\tilde{c})$ of

$$\tilde{c} \in C(\tilde{z}), \qquad \lambda(F_1(\tilde{z},w) + \tilde{c}) + \tilde{z} - z = 0, \tag{1.1.9}$$

then using $\tilde{z}$ to compute an exact solution $(\tilde{w},\tilde{d})$ of

$$\tilde{d} \in D(\tilde{w}), \qquad \lambda(F_2(\tilde{z},\tilde{w}) + \tilde{d}) + \tilde{w} - w = 0, \tag{1.1.10}$$

and finally obtains the next iterate (z^+,w^+) by performing the extragradient step

$$(z^+,w^+) = (z,w) - \lambda(\tilde{v}_1,\tilde{v}_2), \tag{1.1.11}$$

where $\tilde{v}_1 = F_1(\tilde{z},\tilde{w}) + \tilde{c}$ and $\tilde{v}_2 = F_2(\tilde{z},\tilde{w}) + \tilde{d}$. On the other hand, for given $\sigma_1, \sigma_2 \in [0,1]$, an *inexact* instance of the BD-HPE framework computes instead approximate solutions

$(\tilde{z}, \tilde{c}, \varepsilon_1)$ and $(\tilde{w}, \tilde{d}, \varepsilon_2)$ of (1.1.9) and (1.1.10) by relaxing the two inclusions in (1.1.9) and (1.1.10) to

$$\tilde{c} \in C^{\varepsilon_1}(\tilde{z}), \qquad \tilde{d} \in D^{\varepsilon_2}(\tilde{w}),$$

and relaxing the equalities in (1.1.9) and (1.1.10) to the last inequality in the HPE error condition (1.1.6) with $(\tilde{x}, x, \tilde{v}, \varepsilon, \sigma) = (\tilde{z}, z, F_1(\tilde{z}, w) + \tilde{c}, \varepsilon_1, \sigma_1)$ and $(\tilde{x}, x, \tilde{v}, \varepsilon, \sigma) = (\tilde{w}, w, \tilde{v}_2, \varepsilon_2, \sigma_2)$, respectively. It is shown in [39] that, under some suitable assumptions on σ_1, σ_2 and λ, the triple $(\tilde{x}, \tilde{v}, \varepsilon) = ((\tilde{z}, \tilde{w}), (\tilde{v}_1, \tilde{v}_2), \varepsilon_1 + \varepsilon_2)$ satisfies the HPE error condition (1.1.6) with T given by (1.1.8) for some $\sigma \in [0, 1]$, and hence the BD-HPE framework is a special case of the HPE method.

Section 2.1 describes an adaptive block-decomposition HPE (A-BD-HPE) framework similar to the one presented in [39] where we use two different stepsizes λ: one for approximately solving (1.1.9) and (1.1.10); and another one for performing the extragradient step (1.1.11). While the former stepsize is chosen as dictated by the BD-HPE framework of [39], the latter one is chosen as large as possible so as to satisfy the last inequality in the HPE error condition (1.1.6), and thereby improve the practical performance of BD methods.

1.1.3 Overview of existing and proposed methods for large-scale conic SDPs

In this subsection, we give an overview of some existing highly efficient first-order methods for solving (1.1.1) with respect to the following three common operations:

O.1) Evaluation of the linear operator: For given points $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, compute the points $\mathcal{A}x$ and $\mathcal{A}^* y$.

O.2) Projection onto the cone: For a given point $x \in \mathcal{X}$, compute

$$x_K = \arg\min \{\|x - \tilde{x}\| : \tilde{x} \in K\}.$$

O.3) Projection onto the manifold: For a given point $x \in \mathcal{X}$, compute

$$x_{\mathcal{M}} = \arg\min \{\|x - \tilde{x}\| : \tilde{x} \in \mathcal{M}\},$$

where $\mathcal{M} = \{x \in \mathcal{X} : \mathcal{A}x = b\}$. If $\mathcal{A}$ is surjective, it is easy to see that $x_{\mathcal{M}} = x + \mathcal{A}^*(\mathcal{A}\mathcal{A}^*)^{-1}(b - \mathcal{A}x)$. Hence, the cost involved in this operation depends on the ability to solve linear systems of the form $\mathcal{A}\mathcal{A}^* y = \hat{b}$ where $\hat{b} \in \mathcal{Y}$.

Not every first-order method requires all the above operations, but to the best of our knowledge they all perform O.1. Moreover, the low-rank methods in [7, 8, 6] only require O.1. Methods that perform O.2 are commonly known as projection-type methods. To the best of our knowledge the most efficient projection-type methods are the ones that also perform O.3 (e.g., the methods in [30, 35, 33, 67, 68]).

Roughly speaking, a large-scale conic SDP instance as in (1.1.1) and (1.1.4) falls into at least one of the following three categories: i) computation of projections onto the cone K (see O.2) cannot be carried out; ii) large number of constraints m but computation of projections onto the manifold $\mathcal{M}$ (see O.3) can be carried out; and iii) large m but computation of projections onto the manifold $\mathcal{M}$ cannot be carried out. Clearly, any conic SDP instance that falls into category i) is beyond the reach of (second-order) IPMs. Also, for categories ii) and iii), it is assumed that m is large enough so that the instance cannot be solved by IPMs.

For conic SDP instances that fall into category i), the only suitable methods are the low-rank ones as in [7, 8, 6] (and variations thereof) which avoid computing projections onto the cone K by representing the $n_s \times n_s$ symmetric matrix component of the variable x as a low-rank matrix. Projection-type methods such as the ones developed in [30, 35, 33, 67, 68], including the exact BD methods presented in Chapter 2, are the most efficient ones for conic SDP instances that fall into category ii) but not i) since all their main operations, namely O.1, O.2 and O.3, can be carried out. Finally, for conic SDP instances that fall into category iii) but not i), Chapter 3 (see also [34]) presents an inexact BD method that efficiently avoids computation of projections onto the manifold $\mathcal{M}$.

The most challenging conic SDP instances are the ones that fall into category i) and iii) simultaneously and, although one can (often hopelessly) use one of the methods mentioned above to try to solve such instances, there is clearly a need to develop alternative methods which can efficiently solve them. This thesis does not deal with methods for the latter type of instances.

1.1.4 Efficient proximal point BD methods for large-scale conic SDPs

In this subsection we discuss the contribution of Chapters 2 and 3 to proximal point BD methods for solving conic SDP instances that fall into category ii) or iii), but not i) (see Subsection 1.1.3).

Chapter 2 discusses the implementation of two exact first-order block-decomposition instances of the A-BD-HPE framework for solving conic SDP instances that fall into category ii) but not i). Both of these BD instances *exactly* solve the proximal subproblems corresponding to a two-block reformulation of the optimality conditions. The first one (see Section 2.2) is a BD method, namely DSA-BD, for solving standard form conic SDP problems. Numerical results are presented showing that DSA-BD generally outperforms the methods of [30] and [68]. The second one (see Section 2.3) is a BD method, namely 2EBD-HPE, for minimizing the sum of a convex differentiable function with Lipschitz continuous gradient, and two other proper closed convex (possibly, nonsmooth) functions with easily computable resolvents. Numerical results are presented showing that 2EBD-HPE generally outperforms the variant of the BP method in [67], as well as DSA-BD and the methods of [30] and [68], in a benchmark that included the same classes of large-scale graph-related SDP problems tested in [67]. It should be observed though that the implementations in [67] and 2EBD-HPE, are very specific in the sense that they both take advantage of each SDP problem class structure so as to keep the number of variables and/or constraints as small as possible. This contrasts with DSA-BD and the codes described in [30] and [68], which always introduce additional variables and/or constraints into the original SDP formulation to bring it into the required standard form input.

It is worth emphasizing that none of the methods in [30, 35, 33, 67, 68], including the exact BD methods mentioned above, can be used to solve instances that fall into category iii) due to the fact that they all require computation of projections onto the manifold $\mathcal{M}$ and/or solutions of linear systems with coefficient matrix related to $\mathcal{A}\mathcal{A}^*$. Even though there exist exact BD methods (see for example the variant of the method in [35] with $\mathcal{U} = \mathcal{I}$) that avoid computation of projections onto the manifold $\mathcal{M}$ and, as a consequence, can be used to solve instances that fall into category iii) but not i), we have observed that they

generally do not perform well computationally.

Chapter 3 presents an inexact BD method that avoids computation of projections onto the manifold $\mathcal{M}$ and is highly efficient for conic SDP instances that fall into category iii) but not i). More specifically, we present a BD method, namely CG-ISBD, that instead of exactly solving linear systems of the form $\mathcal{A}\mathcal{A}^*y = \hat{b}$, computes *inexact* solutions of augmented primal-dual linear systems satisfying a certain relative error condition. The latter relative error condition naturally appears as part of the HPE error condition which in turn guarantees the overall convergence of the BD method. Our implementation of CG-ISBD obtains the aforementioned inexact solutions of the augmented linear systems by applying the conjugate gradient (CG) method to linear systems of the form $(\mathcal{I}+\alpha\mathcal{A}\mathcal{A}^*)y = \hat{b}$, where $\mathcal{I}$ is the identity operator and α is a positive scalar.

We present numerical results showing the ability of CG-ISBD to solve various conic SDP instances of the form (1.1.1) and (1.1.4) for which the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3) is prohibitively expensive, and as a result cannot be handled by the methods in Chapter 2 as well as the ones in [30, 35, 33, 67, 68]. In these numerical results, we also show that our method substantially outperforms the latest implementation (see [6]) of SDPLR introduced in [7, 8]. SDPLR is a first-order augmented Lagrangian method applied to a nonlinear reformulation of the original problem (1.1.1) which restricts the $n_s \times n_s$ symmetric matrix component of the variable x to a low-rank matrix as mentioned above. Even though there are other first-order methods that avoid projecting onto the manifold $\mathcal{M}$ (see for example the BD variant in [35] with $\mathcal{U} = \mathcal{I}$), to the best of our knowledge, SDPLR is computationally the most efficient one.

Finally, it should be mentioned that although the BD methods described in [39] are simple and have nice convergence properties, their implementation in its pure form is far from being efficient. Chapters 2 and 3 introduce several ingredients to the BD methods of [39] to obtain highly efficient algorithms for solving large-scale conic SDPs of the form (1.1.1) and (1.1.4). The first ingredient is the use of an aggressive choice of stepsize based on a certain error criterion for performing the extragradient step (1.1.7). The second important idea is to endow the spaces (e.g. $\mathcal{X}$ and $\mathcal{Y}$) with scaled inner products for the implementation

of the methods. The third idea is to allow the scaled inner products to dynamically change as the algorithm progresses, with the aim of properly balancing the sizes of the error residuals of each block so as to make them go to zero according to the same order of magnitude. Finally, the fourth idea is to properly choose the initial parameters of the scaled inner products before starting to iterate the methods.

1.2 Large-scale convex optimization

We consider the convex optimization problem

$$\phi^* := \min_{x \in \mathcal{X}} \phi(x), \tag{1.2.1}$$

where the following conditions are assumed:

C.1) $\phi : \mathcal{X} \to [-\infty, \infty]$ is a proper closed convex function;

C.2) the set X^* of optimal solutions of (1.2.1) is non-empty.

For many large-scale convex optimization problems, when accuracy requirements are not high, first-order methods are methods of choice due to their cheap iteration cost and global convergence properties. In his seminal paper [42] (see also [45]), Nesterov presented a scheme for accelerating first-order methods for (1.2.1), more specifically, the steepest descent method for differentiable ϕ and the projected gradient method for

$$\phi = f + \delta_C,$$

where f is differentiable on the closed convex set C and δ_C is defined as in (1.4.2) (i.e., constrained convex optimization problems). He shows that the rate of convergence of these methods, namely $\mathcal{O}(1/k)$, where k denotes the iteration count, can be improved to $\mathcal{O}(1/k^2)$ by considering their accelerated variants. Due to its wide use for solving large-scale convex optimization problems arising in several applications, other accelerated variants of Nesterov's method (see for example [1, 13, 19, 25, 42, 43, 47, 46, 65]) have been proposed and studied in the literature. In addition, interest into these methods has been recently renewed with the development of smoothing techniques for non-smooth convex problems

(see for example [14, 43, 45, 44, 48]), where accelerated methods are used to minimize a smooth approximations of non-smooth objective functions.

There has been an increasing concern of improving the practical performance of these methods for the solution of large-scale convex optimization problems. Recently, a variant introduced in [1] reports promising numerical results showing its ability to solve ℓ_1-regularized least squares problems. Also, variants like the ones in [20, 19] fine tunes Nesterov's method to reduce the number of iterations at the cost of inexact projected line searches.

In Chapter *4*, we present a new adaptive accelerated variant of Nesterov's method, namely the AA method, for solving a class of convex optimization problems as in (1.2.1), in which certain acceleration parameters are adaptively (and aggressively) chosen so as to: 1) preserve the theoretical iteration-complexity of the original method; and 2) substantially improve its practical performance in comparison to the other existing variants. We also present a generic accelerated framework for solving (1.2.1) and derive two bounds on the functional gap, namely: one in terms of the sequence of aggregated acceleration parameters and another in terms of the sequence of (possibly variable) stepsizes. In addition, we propose a specific adaptive choice of the sequence of aggregated acceleration parameters whose goal is to greedily minimize the derived bound on the functional gap in terms of these parameters. The AA method is a special instance of the generic accelerated framework for a special class of convex programming problems whose objective function ϕ has the additional property that, at each point $\bar{x}$, one can find proper closed convex $g_{\bar{x}}$ with easily computable resolvent such that $g_{\bar{x}} \leq \phi \leq g_{\bar{x}} + L\|\cdot - \bar{x}\|^2/2$, where $L \geq 0$ is a constant dependent on ϕ only. An example of such a function ϕ consists of the sum of a function with Lipschitz continuous gradient and an easy nonsmooth proper closed convex function. Another example consists of the composition of a convex non-decreasing function on $\mathbb{R}^m$ with a tuple of m functions as above. Numerical results are presented demonstrating that the proposed adaptive accelerated method performs quite well compared to other existing variants proposed earlier in the literature.

1.3 Organization of the thesis

This thesis is organized as follows.

In Chapter 2, we consider *exact BD methods* for conic programming. First, Section 2.1 presents the A-BD-HPE framework, and corresponding iteration-complexity results, in the context of a block-structured monotone inclusion problem. Then, Section 2.2 introduces an exact BD method for standard form conic programming problems. On the other hand, Section 2.3 considers an exact BD method for composite convex optimization with applications to conic programming that have the ability to take advantage of the original problem class structure by directly handling non-standard form formulations. Section 2.4 presents some final remarks.

In Chapter 3, we consider *inexact BD methods* for conic programming. First, in Section 3.1 we present an inexact first-order BD method for solving the standard form conic programming problem (1.1.1) which avoids the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1.1). Then, Section 3.2 describes a practical variant of this latter inexact BD method which incorporates a new dynamic scaling scheme and the use of the CG method to inexactly solve augmented primal-dual linear systems. In Section 3.3, we present computational results showing the efficiency of our inexact BD method for solving various extra large SDP instances, several of which cannot be solved by other existing methods. Section 3.4 presents some final remarks.

In Chapter 4, we consider *adaptive accelerated gradient methods* for a general class of convex optimization problems. First, Section 4.1 presents a generic accelerated framework, namely the GenA framework, for convex optimization (1.2.1). Then, Section 4.2 introduces a scheme that adaptively and aggressively chooses certain acceleration parameters of the GenA framework in order to substantially improve its practical performance without compromising its theoretical iteration-complexity. Section 4.3 introduces a first-order instance of the GenA framework that incorporates the latter scheme. Numerical results demonstrating that the proposed adaptive accelerated method performs quite well compared to other variants proposed earlier in the literature are reported in Section 4.4. Section 4.5 presents some final remarks.

1.4 Notation

In this section, we introduce some notation used throughout this thesis.

Let $\mathbb{R}$ denote the set of real numbers, $\bar{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ denote the set of extended real numbers, $\mathbb{R}^n$ denote the n-dimensional Euclidean space, $\mathbb{R}^n_+$ denote the cone of nonnegative vectors in $\mathbb{R}^n$, $\mathcal{S}^n$ denote the set of all $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ denote the cone of $n \times n$ symmetric positive semidefinite matrices. Throughout this thesis, we let $\mathcal{X}$ and $\mathcal{Y}$ be finite dimensional inner product spaces with inner products and associated norms denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively. The canonical norm of the pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is defined as

$$\|(x, y)\| := \sqrt{\|x\|^2 + \|y\|^2}.$$

Let a closed convex set $C \subseteq \mathcal{X}$ be given. The *projection* operator $\Pi_C : \mathcal{X} \to C$ onto C is defined by

$$\Pi_C(x) = \arg\min_{\tilde{x} \in C} \{\|x - \tilde{x}\|\} \qquad \forall x \in \mathcal{X},$$

and the *distance function* $\mathrm{dist}_C : \mathcal{X} \to \mathbb{R}_+$ with respect to C is defined as

$$\mathrm{dist}_C(x) := \min_{\tilde{x} \in C} \{\|x - \tilde{x}\|\}, \qquad \forall x \in \mathcal{X}. \tag{1.4.1}$$

The *indicator function* of C is the function $\delta_C : \mathcal{X} \to \bar{\mathbb{R}}$ defined as

$$\delta_C(x) = \begin{cases} 0, & x \in C, \\ \infty, & \text{otherwise} \end{cases} \tag{1.4.2}$$

and the *normal cone* operator $N_C : \mathcal{X} \rightrightarrows \mathcal{X}$ of C is the point-to-set map given by

$$N_C(x) = \begin{cases} \emptyset, & x \notin C, \\ \{w \in \mathcal{X} : \langle \tilde{x} - x, w \rangle \leq 0, \ \forall \tilde{x} \in C\}, & x \in C. \end{cases} \tag{1.4.3}$$

The *induced norm* $\|\mathcal{A}\|$ of a linear operator $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is defined as

$$\|\mathcal{A}\| := \max_{x \in \mathcal{X}} \{\|\mathcal{A}x\| : \|x\| \leq 1\}.$$

Moreover, if $\mathcal{A}$ is invertible, the *condition number* $\kappa(\mathcal{A})$ of $\mathcal{A}$ is defined as

$$\kappa(\mathcal{A}) := \|\mathcal{A}\| \|\mathcal{A}^{-1}\|.$$

If $\mathcal{X} = \mathcal{Y}$, we denote by $\lambda_{\max}(\mathcal{A})$ and $\lambda_{\min}(\mathcal{A})$ the maximum and minimum eigenvalues of $\mathcal{A}$, respectively. If $\mathcal{A}$ is identified with a matrix, $\mathrm{nnz}(\mathcal{A})$ denotes the number of non-zeros of $\mathcal{A}$.

The domain of a point-to-point function F is denoted by $\mathrm{Dom}\, F$, and the effective domain of a function $f : \mathcal{X} \to \bar{R}$ is defined as

$$\mathrm{dom}\, f := \{x \in \mathcal{X} : f(x) < \infty\}\,.$$

Chapter II

EXACT BLOCK-DECOMPOSITION METHODS FOR CONIC PROGRAMMING

In this chapter, we introduce two exact BD methods for conic programming. This chapter is divided in four sections. First, we discuss an adaptive BD framework for solving block-structured monotone inclusion problems in Section 2.1. Section 2.2 presents an exact BD method for standard form conic programming problems as in (1.1.1) and reports numerical results showing that it generally outperforms the methods of [30] and [68]. Section 2.3 presents an exact BD method for minimizing the sum of a convex differentiable function with Lipschitz continuous gradient, and two other proper closed convex (possibly, nonsmooth) functions with easily computable resolvents. In addition, the latter method is specialized to the context of conic programming and numerical results are presented showing that it generally outperforms the variant of the BP method in [67]. Finally, Section 2.4 presents some final remarks.

2.1 An adaptive block-decomposition framework

In this section, we discuss an adaptive block-decomposition HPE (A-BD-HPE) framework which is an extension of the BD-HPE framework introduced in [39]. This framework is analyzed in the context of a block-structured monotone inclusion problem similar to the one in Section 3 of [39], but with the addition of an adaptive (and aggressive) stepsize choice for performing the extragradient step. This section is divided into two parts. The first one reviews some basic definitions and facts about ε-subdifferentials of functions and ε-enlargements of monotone operators. The second one presents the A-BD-HPE framework.

2.1.1 The ε-subdifferential and ε-enlargement of monotone operators

Let $\mathcal{Z}$ denote a finite dimensional inner product space. A point-to-set operator $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is a relation $T \subset \mathcal{Z} \times \mathcal{Z}$ and

$$T(z) := \{v \in \mathcal{Z} : (z, v) \in T\}.$$

Alternatively, one can consider T as a multi-valued function of $\mathcal{Z}$ into the family $\wp(\mathcal{Z}) = 2^{(\mathcal{Z})}$ of subsets of $\mathcal{Z}$. Regardless of the approach, it is usual to identify T with its graph defined as

$$Gr(T) := \{(z, v) \in \mathcal{Z} \times \mathcal{Z} : v \in T(z)\}.$$

The domain of T , denoted by $\operatorname{Dom} T$, is defined as

$$\operatorname{Dom} T := \{z \in \mathcal{Z} : T(z) \neq \emptyset\}.$$

An operator $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is *affine* if its graph is an affine manifold. Moreover, $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ is monotone if

$$\langle v - \tilde{v}, z - \tilde{z} \rangle \geq 0, \qquad \forall (z, v), (\tilde{z}, \tilde{v}) \in Gr(T),$$

and T is maximal monotone if it is monotone and maximal in the family of monotone operators with respect to the partial order of inclusion, i.e., $S : \mathcal{Z} \rightrightarrows \mathcal{Z}$ monotone and $Gr(S) \supset Gr(T)$ implies that $S = T$.

Lemma 2.1 (Moreau's identity; see Lemma 6.3 in [39])**.** *Let $\lambda > 0$, $z \in \mathcal{Z}$ and $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ be a point to set maximal monotone operator. Then,*

$$z = (I + \lambda T)^{-1}(z) + \lambda \left(I + \lambda^{-1} T^{-1}\right)^{-1} \left(\lambda^{-1} z\right).$$

In [4], Burachik, Iusem and Svaiter introduced the ε-enlargement of maximal monotone operators. In [36] this concept was extended to a generic point-to-set operator in $\mathcal{Z}$ as follows. Given $T : \mathcal{Z} \rightrightarrows \mathcal{Z}$ and a scalar ε, define $T^{\varepsilon} : \mathcal{Z} \rightrightarrows \mathcal{Z}$ as

$$T^{\varepsilon}(z) := \{v \in \mathcal{Z} : \langle z - \tilde{z}, v - \tilde{v} \rangle \geq -\varepsilon, \forall \tilde{z} \in \mathcal{Z}, \forall \tilde{v} \in T(\tilde{z})\}, \quad \forall z \in \mathcal{Z}.$$

We now state a few useful properties of the operator T^{ε} that will be needed in our presentation.

Proposition 2.2. *Let $T, T' : \mathcal{Z} \rightrightarrows \mathcal{Z}$. Then,*

a) if $\varepsilon_1 \leq \varepsilon_2$, then $T^{\varepsilon_1}(z) \subseteq T^{\varepsilon_2}(z)$ for every $z \in \mathcal{Z}$;

b) $T^{\varepsilon}(z) + (T')^{\varepsilon'}(z) \subseteq (T + T')^{\varepsilon+\varepsilon'}(z)$ for every $z \in \mathcal{Z}$ and $\varepsilon, \varepsilon' \in \mathbb{R}$;

c) T is monotone if and only if $T \subseteq T^0$;

d) T is maximal monotone if and only if $T = T^0$;

The following result, whose proof can be found in Lemma 3.3 in [36], gives the characterization of the ε-enlargement of the normal cone of a closed convex cone.

Lemma 2.3 (Lemma 3.3 in [36])**.** *If $K \subseteq \mathcal{Z}$ is a nonempty closed convex cone and K^* is its dual cone defined in* (1.1.3)*, then for every $x \in K$ and $q \in \mathcal{Z}$, we have*

$$-q \in (N_K)^{\varepsilon}(x) \iff q \in K^*, \ \langle x, q \rangle \leq \varepsilon.$$

For a scalar $\varepsilon \geq 0$, the ε-subdifferential of a function $f : \mathcal{X} \to \bar{\mathbb{R}}$ is the operator $\partial_\varepsilon f : \mathcal{X} \rightrightarrows \mathcal{X}$ defined as

$$\partial_\varepsilon f(x) := \{w \in \mathcal{X} : f(\tilde{x}) \geq f(x) + \langle \tilde{x} - x, w \rangle - \varepsilon, \forall \tilde{x} \in \mathcal{X}\}, \quad \forall x \in \mathcal{X}.$$

When $\varepsilon = 0$, the operator $\partial_\varepsilon f$ is simply denoted by ∂f and is referred to as the subdifferential of f. The operator ∂f is trivially monotone if f is proper. If f is a proper lower semi-continuous convex function, then ∂f is maximal monotone [55].

The conjugate f^* of f is the function $f^* : \mathcal{Z} \to [-\infty, \infty]$ defined as

$$f^*(v) = \sup_{z \in \mathcal{Z}} \langle v, z \rangle_{\mathcal{Z}} - f(z), \quad \forall v \in \mathcal{Z}.$$

The following result lists some useful properties about the ε-subdifferential of a proper convex function.

Proposition 2.4. *Let $f : \mathcal{Z} \to (-\infty, \infty]$ be a proper convex function. Then,*

a) $\partial_\varepsilon f(z) \subseteq (\partial f)^{\varepsilon}(z)$ for any $\varepsilon \geq 0$ and $z \in \mathcal{Z}$;

b) if f is closed, then $\partial(f^*) = (\partial f)^{-1}$;

c) if $v \in \partial f(z)$ and $f(\tilde{z}) < \infty$, then $v \in \partial_\varepsilon f(\tilde{z})$, for every $\varepsilon \geq f(\tilde{z}) - [f(z) + \langle \tilde{z} - z, v\rangle]$.

For a closed convex cone $K \subseteq \mathcal{Z}$, we have the following characterization about the ε-subdiferential of δ_K.

Proposition 2.5. *Let $K \subseteq \mathcal{Z}$ be a (nonempty) closed convex cone. Then, for any $\varepsilon \geq 0$, the pair $(z, w) \in \mathcal{Z} \times \mathcal{Z}$ satisfies $w \in -\partial_\varepsilon \delta_K(z)$ if and only if $z \in K$, $w \in K^*$ and $\langle z, w\rangle_{\mathcal{Z}} \leq \varepsilon$, where K^* is dual cone of K .*

Finally, we refer the reader to [5, 63] for further discussion on the ε-enlargement of a maximal monotone operator.

2.1.2 The A-BD-HPE framework

We now discuss the A-BD-HPE framework which is an extension of the BD-HPE framework introduced in [39]. This framework is analyzed in the context of a block-structured monotone inclusion problem similar to the one in Section 3 of [39]. In what follows we give the details of this block-structured monotone inclusion problem.

Let $\mathcal{Z}$ and $\mathcal{W}$ be finite dimensional inner product spaces with associated inner products denoted by $\langle \cdot, \cdot \rangle_{\mathcal{Z}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{W}}$, respectively, and associated norms denoted by $\| \cdot \|_{\mathcal{Z}}$ and $\| \cdot \|_{\mathcal{W}}$, respectively. We endow the product space $\mathcal{Z} \times \mathcal{W}$ with the canonical inner product $\langle \cdot, \cdot \rangle_{\mathcal{Z},\mathcal{W}}$ and associated canonical norm $\| \cdot \|_{\mathcal{Z}\times\mathcal{W}}$ defined as

$$\langle (z, w), (z', w') \rangle_{\mathcal{Z}\times\mathcal{W}} := \langle z, z' \rangle_{\mathcal{Z}} + \langle w, w' \rangle_{\mathcal{W}}, \quad \|(z, w)\|_{\mathcal{Z}\times\mathcal{W}} := \sqrt{\langle (z, w), (z, w) \rangle_{\mathcal{Z}\times\mathcal{W}}}, \tag{2.1.1}$$

for all $(z, w), (z', w') \in \mathcal{Z} \times \mathcal{W}$. Our problem of interest in this section is the monotone inclusion problem of finding $(z, w) \in \mathcal{Z} \times \mathcal{W}$ such that

$$(0, 0) \in [F + (C \otimes D)](z, w), \tag{2.1.2}$$

where

$$F(z, w) = (F_1(z, w), F_2(z, w)) \in \mathcal{Z} \times \mathcal{W}, \qquad (C \otimes D)(z, w) = C(z) \times D(w) \subseteq \mathcal{Z} \times \mathcal{W}$$

and the following conditions are assumed:

A.1) $C : \mathcal{Z} \rightrightarrows \mathcal{Z}$ and $D : \mathcal{W} \rightrightarrows \mathcal{W}$ are maximal monotone operators (with respect to $\langle \cdot, \cdot \rangle_{\mathcal{Z}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{W}}$, respectively);

A.2) $F : \mathrm{Dom}\, F \subseteq \mathcal{Z} \times \mathcal{W} \to \mathcal{Z} \times \mathcal{W}$ is a continuous map such that $\mathrm{Dom}\, F \supseteq \mathcal{Z} \times \mathrm{cl}\,(\mathrm{Dom}\, D)$;

A.3) F is monotone on $\mathrm{cl}\,(\mathrm{Dom}\ C) \times \mathrm{cl}\,(\mathrm{Dom}\ D)$; (with respect to $\langle (\cdot, \cdot), (\cdot, \cdot) \rangle_{\mathcal{Z},\mathcal{W}}$);

A.4) there exists $L > 0$ such that

$$\|F_1(z, w') - F_1(z, w)\|_{\mathcal{Z}} \leq L\|w' - w\|_{\mathcal{W}}, \quad \forall z \in \mathrm{Dom}\ C, \ \ \forall w, w' \in \mathcal{W}.$$

Assumption A.1 implies that $C \otimes D$ is maximal monotone with respect to $\langle (\cdot, \cdot), (\cdot, \cdot) \rangle_{\mathcal{Z},\mathcal{W}}$. Hence, in view of Assumption A.2 above and Proposition A.1 of [37], it follows that the operator $F + C \otimes D$ in (2.1.2) is maximal monotone. Note that problem (2.1.2) is equivalent to

$$0 \in F_1(z, w) + C(z), \qquad 0 \in F_2(z, w) + D(w).$$

We now state an extension of the BD-HPE framework of [39] which uses an adaptive rule for aggressively choosing the extragradient stepsize.

A-BD-HPE Framework: An adaptive block-decomposition HPE framework

0) Let $(z^0, w^0) \in \mathcal{Z} \times \mathcal{W}$, $\sigma \in (0,1]$, $\sigma_z \in [0,1]$ and $\tilde{\sigma}_z, \sigma_w \in [0,1)$ be given and set $k = 1$;

1) choose $\tilde{\lambda}_k > 0$ such that

$$\sigma_k := \lambda_{\max}\left(\begin{bmatrix} \sigma_z^2 & \tilde{\lambda}_k \tilde{\sigma}_z L \\ \tilde{\lambda}_k \tilde{\sigma}_z L & \sigma_w^2 + \tilde{\lambda}_k^2 L^2 \end{bmatrix}\right)^{1/2} \leq \sigma; \tag{2.1.3}$$

2) compute $\tilde{z}^k, \tilde{c}^k \in \mathcal{Z}$ and $\varepsilon_k^z \geq 0$ such that

$$\tilde{c}^k \in C^{\varepsilon_k^z}(\tilde{z}^k), \quad \|\tilde{\lambda}_k[F_1(\tilde{z}^k, w^{k-1}) + \tilde{c}^k] + \tilde{z}^k - z^{k-1}\|_{\mathcal{Z}}^2 + 2\tilde{\lambda}_k \varepsilon_k^z \leq \sigma_z^2 \|\tilde{z}^k - z^{k-1}\|_{\mathcal{Z}}^2, \tag{2.1.4}$$

$$\|\tilde{\lambda}_k[F_1(\tilde{z}^k, w^{k-1}) + \tilde{c}^k] + \tilde{z}^k - z^{k-1}\|_{\mathcal{Z}} \leq \tilde{\sigma}_z \|\tilde{z}^k - z^{k-1}\|_{\mathcal{Z}}; \tag{2.1.5}$$

3) compute $\tilde{w}^k, \tilde{d}^k \in \mathcal{W}$ and $\varepsilon_k^w \geq 0$ such that

$$\tilde{d}^k \in D^{\varepsilon_k^w}(\tilde{w}^k), \quad \|\tilde{\lambda}_k[F_2(\tilde{z}^k, \tilde{w}^k) + \tilde{d}^k] + \tilde{w}^k - w^{k-1}\|_{\mathcal{W}}^2 + 2\tilde{\lambda}_k \varepsilon_k^w \leq \sigma_w^2 \|\tilde{w}^k - w^{k-1}\|_{\mathcal{W}}^2; \tag{2.1.6}$$

4) choose λ_k to be the largest $\lambda > 0$ such that

$$\begin{aligned} \|\lambda[F(\tilde{z}^k, \tilde{w}^k) + (\tilde{c}^k, \tilde{d}^k)] + (\tilde{z}^k, \tilde{w}^k) - (z^{k-1}, w^{k-1})\|_{\mathcal{Z}\times\mathcal{W}}^2 + 2\lambda(\varepsilon_k^z + \varepsilon_k^w) \leq \\ \sigma^2 \|(\tilde{z}^k, \tilde{w}^k) - (z^{k-1}, w^{k-1})\|_{\mathcal{Z}\times\mathcal{W}}^2; \end{aligned} \tag{2.1.7}$$

5) set

$$(z^k, w^k) = (z^{k-1}, w^{k-1}) - \lambda_k[F(\tilde{z}^k, \tilde{w}_k) + (\tilde{c}^k, \tilde{d}^k)], \tag{2.1.8}$$

$k \leftarrow k + 1$, and go to step 1.

The A-BD-HPE framework is a more aggressive version of the BD-HPE framework studied in [39]. In contrast to the A-BD-HPE framework which chooses the extragradient stepsize as the largest scalar satisfying (2.1.7), the BD-HPE framework in [39] performs the extragradient step with $\lambda_k = \tilde{\lambda}_k$. The following result shows that $\tilde{\lambda}_k$ satisfies (2.1.7) and, as a consequence, guarantees the well-definedness of the adaptive extragradient stepsize λ_k in step 3 of the A-BD-HPE framework.

Proposition 2.6 (Proposition 2.2 of [35]). *Consider the sequences* $\{(z^k, w^k)\}$, $\{(\tilde{z}^k, \tilde{w}^k)\}$, $\{(\tilde{c}^k, \tilde{d}^k)\}$, $\{\tilde{\lambda}_k\}$ *and* $\{(\varepsilon_k^z, \varepsilon_k^w)\}$ *generated by the A-BD-HPE framework. Then, for every* $k \in \mathbb{N}$,

$$F(\tilde{z}^k, \tilde{w}_k) + (\tilde{c}^k, \tilde{d}^k) \in [F + (C \otimes D)^{\varepsilon_k^z + \varepsilon_k^w}](\tilde{z}^k, \tilde{w}_k) \subset [F + (C \otimes D)]^{\varepsilon_k^z + \varepsilon_k^w}(\tilde{z}^k, \tilde{w}_k) \quad (2.1.9)$$

and $\lambda = \tilde{\lambda}_k$ *satisfies* (2.1.7). *As a consequence* $\lambda_k \geq \tilde{\lambda}_k$.

Proof. This result follows from Proposition 3.1 in [39]. □

In view of (2.1.7), (1.1.7) and (2.1.9), it follows that the A-BD-HPE framework is a special case of the HPE framework for solving (2.1.2). This observation allows us to obtain complexity results for the A-BD-HPE framework using the general complexity results derived in [36]. In what follows, we state two convergence results whose proofs are analogous to those of Theorems 3.2 and 3.3 of [39], but use Proposition 2.6 above instead of Proposition 3.1 in [39].

Theorem 2.7 (Theorem 2.3 of [35]). *Assume that* $\sigma < 1$ *and consider the sequences* $\{(\tilde{z}^k, \tilde{w}^k)\}$, $\{(\tilde{c}^k, \tilde{d}^k)\}$, $\{\lambda_k\}$ *and* $\{(\varepsilon_k^z, \varepsilon_k^w)\}$ *generated by the A-BD-HPE framework and let* d_0 *denote the distance of the initial point* $(z^0, w^0) \in \mathcal{Z} \times \mathcal{W}$ *to the solution set of* (2.1.2) *with respect to* $\|(\cdot, \cdot)\|_{\mathcal{Z},\mathcal{W}}$. *Then, for every* $\alpha \in \mathbb{R}$ *and* $k \in \mathbb{N}$,

$$(\tilde{c}^k, \tilde{d}^k) \in C^{\varepsilon_k^z}(\tilde{z}^k) \times D^{\varepsilon_k^w}(\tilde{w}^k),$$

and there exists $i \leq k$ *such that*

$$\|F(\tilde{z}^i, \tilde{w}^i) + (\tilde{c}^i, \tilde{d}^i)\|_{\mathcal{Z}\times\mathcal{W}} \leq d_0 \sqrt{\frac{(1+\sigma)}{(1-\sigma)}\left(\frac{\lambda_i^{\alpha-2}}{\sum_{j=1}^k \lambda_j^\alpha}\right)}, \quad \varepsilon_i^z + \varepsilon_i^w \leq \frac{d_0^2\sigma^2}{2(1-\sigma^2)}\left(\frac{\lambda_i^{\alpha-1}}{\sum_{j=1}^k \lambda_j^\alpha}\right).$$

In particular, for every $k \in \mathbb{N}$, *there exists* $i \leq k$ *such that*

$$\|F(\tilde{z}^i, \tilde{w}^i) + (\tilde{c}^i, \tilde{d}^i)\|_{\mathcal{Z}\times\mathcal{W}} \leq d_0 \sqrt{\frac{1+\sigma}{1-\sigma}\left(\frac{1}{\lambda_i \sum_{j=1}^k \lambda_j}\right)}, \quad \varepsilon_i^z + \varepsilon_i^w \leq \frac{\sigma^2 d_0^2}{2(1-\sigma^2)\sum_{j=1}^k \lambda_j}.$$

Theorem 2.8 (Theorem 2.4 of [35]). *Assume that* F *is affine and consider the sequences* $\{(\tilde{z}^k, \tilde{w}^k)\}$, $\{(\tilde{c}^k, \tilde{d}^k)\}$, $\{\lambda_k\}$ *and* $\{(\varepsilon_k^z, \varepsilon_k^w)\}$ *generated by the A-BD-HPE framework and define for every* $k \in \mathbb{N}$:

$$(\tilde{z}^{a,k}, \tilde{w}^{a,k}) = \frac{1}{\Lambda_k}\sum_{i=1}^k \lambda_i(\tilde{z}^i, \tilde{w}^i), \quad (\tilde{c}^{a,k}, \tilde{d}^{a,k}) = \frac{1}{\Lambda_k}\sum_{i=1}^k \lambda_i(\tilde{c}^i, \tilde{d}^i), \quad (2.1.10)$$

and

$$\varepsilon_k^{z,a} := \frac{1}{\Lambda_k}\sum_{i=1}^{k}\lambda_i(\varepsilon_i^z + \langle \tilde{z}^i - \tilde{z}^{a,k}, \tilde{c}^i\rangle) \geq 0, \qquad \varepsilon_k^{w,a} := \frac{1}{\Lambda_k}\sum_{i=1}^{k}\lambda_i(\varepsilon_i^w + \langle \tilde{w}^i - \tilde{w}^{a,k}, \tilde{d}^i\rangle) \geq 0,$$

where $\Lambda_k = \sum_{i=1}^{k}\lambda_i$. *Let* d_0 *denote the distance of the initial point* $(z^0, w^0) \in \mathcal{Z}\times\mathcal{W}$ *to the solution set of* (2.1.2) *with respect to* $\|(\cdot,\cdot)\|_{\mathcal{Z},\mathcal{W}}$ *and* $\sigma_{zw} = \max\{\sigma_z, \sigma_w\}$. *Then, for every* $k \in \mathbb{N}$,

$$(\tilde{c}^{a,k}, \tilde{d}^{a,k}) \in C^{\varepsilon_k^{z,a}}(\tilde{z}^{a,k}) \times D^{\varepsilon_k^{w,a}}(\tilde{w}^{a,k}), \quad \left\|F(\tilde{z}^{a,k}, \tilde{w}^{a,k}) + (\tilde{c}^{a,k}, \tilde{d}^{a,k})\right\|_{\mathcal{Z}\times\mathcal{W}} \leq \frac{2d_0}{\Lambda_k},$$

$$\varepsilon_k^{z,a} + \varepsilon_k^{w,a} \leq \frac{2d_0^2}{\Lambda_k}(1+\bar{\eta}_k),$$

where

$$\bar{\eta}_k := \frac{2\sqrt{2}\sigma}{1-\sigma_{zw}}\left(1 + \frac{1}{(1-\sigma_w)^2}\right)^{1/2}.$$

Observe that the convergence rate bounds described in Theorems 2.7 (with $\alpha = 1$) and 2.8 suggest that the rate of convergence of the A-BD-HPE framework becomes better the larger the stepsize λ_k is chosen (under the condition that (2.1.7) is satisfied so as to guarantee that Theorems 2.7 and 2.8 still apply). In fact, the choice of λ_k at step 3 of the A-BD-HPE framework is motivated by this observation.

2.2 An exact BD method for standard form conic SDP

In this section, we introduce an exact BD method for standard form conic programming problems as in (1.1.1). An instance of the A-BD-HPE framework for solving the conic programming problem (1.1.1) in which the $\mathcal{Z}$ space is endowed with a scaled inner product is described in Subsection 2.2.1. Subsection 2.2.2 describes in detail all the ingredients needed to speed-up the pure form of the adaptive block-decomposition HPE method, and presents numerical results demonstrating the efficiency of the resulting algorithm for solving many large instances of (1.1.1) and (1.1.4).

2.2.1 A scaled A-BD method for conic programming

In this section, we introduce an instance of the A-BD-HPE framework described in Section 2.1 applied to the standard form conic problem (1.1.1), and defines $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$ as

scaled inner products constructed by means of the original inner products $\langle\cdot,\cdot\rangle$ of $\mathcal{X}$ and $\mathcal{Y}$. (Recall from Section 1.4 that the original inner products in $\mathcal{X}$ and $\mathcal{Y}$ used in (1.1.1) are both being denoted by $\langle\cdot,\cdot\rangle$.) We will use the convergence results of Section 2.1 to give a plausible argument showing that an appropriate choice of scaled inner product in the $\mathcal{Z}$ space may lead to a substantial speed-up of the BD method relative to its unscaled version.

We consider problem (1.1.1) with the following assumptions:

D.1) $\mathcal{A}:\mathcal{X}\to\mathcal{Y}$ is a surjective linear map and $b\in\mathcal{Y}$;

D.2) there exists $x^*\in\mathcal{X}$ satisfying the inclusion

$$0\in c+\partial\delta_K(x)+N_{\mathcal{M}}(x), \tag{2.2.1}$$

where $\mathcal{M}:=\{x\in\mathcal{X}:\mathcal{A}(x)=b\}$.

We now make a few observations about the above assumptions. First, any x^* as in D.2 is an optimal solution of (1.1.1). Second, observe that a sufficient condition for (2.2.1) to hold is that (1.1.1) has an optimal solution and satisfies the Slater condition, i.e. $\mathcal{A}\hat{x}=b$ for some $\hat{x}\in\mathrm{ri}(K)$. Third, (2.2.1) is equivalent to the existence of $y^*\in\mathcal{Y}$ such that the pair (x^*,y^*) satisfies the inclusion

$$\mathcal{A}x-b=0,\qquad c+\partial\delta_K(x)-\mathcal{A}^*y\ni 0. \tag{2.2.2}$$

Fourth, the set of solutions of the above inclusion is exactly $X^*\times Y^*$, where X^* and Y^* denote the set of optimal solutions of (1.1.1) and (1.1.2), respectively. Fifth, observe that (2.2.2) can be easily put into the form (2.1.2) and that assumptions A.1–A.4 all hold when $\mathcal{Z}=\mathcal{Y}$, $\mathcal{W}=\mathcal{X}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}=\langle\cdot,\cdot\rangle$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}=\langle\cdot,\cdot\rangle$, where $\langle\cdot,\cdot\rangle$ denotes the original inner products of $\mathcal{X}$ and $\mathcal{Y}$. Hence, one can apply any instance of the A-BD-HPE framework to solve (2.2.2).

However, from the computational point of view, it is more efficient to introduce a scaled inner product in the $\mathcal{Z}$ space and work with a scaled version of (2.2.2). More specifically, given a self adjoint positive definite linear mapping $\mathcal{U}:\mathcal{Y}\to\mathcal{Y}$ and letting $\mathcal{Z}=\mathcal{Y}$ and $\mathcal{W}=\mathcal{X}$, endow $\mathcal{Z}$ and $\mathcal{W}$ with the inner products defined as

$$\langle\cdot,\cdot\rangle_{\mathcal{Z}}:=\langle\cdot,\mathcal{U}\cdot\rangle,\qquad\langle\cdot,\cdot\rangle_{\mathcal{W}}:=\langle\cdot,\cdot\rangle, \tag{2.2.3}$$

respectively, and define F, C and D as

$$F(x,y) := \begin{pmatrix} \mathcal{U}^{-1}(\mathcal{A}x - b) \\ c - \mathcal{A}^* y \end{pmatrix}, \quad C(x) = 0, \quad D(y) = \partial\delta_K(x) = N_K(x), \tag{2.2.4}$$

where the normal cone $N_K(x)$ is with respect to $\langle\cdot,\cdot\rangle_{\mathcal{W}} := \langle\cdot,\cdot\rangle$.

The following proposition can be easily shown.

Proposition 2.9. *F, C and D defined in* (2.2.4) *and the above inner products defined in* (2.2.3) *satisfy assumptions A.1–A.4 of Section 2.1 with* $L = \left\|\mathcal{U}^{-1/2}\mathcal{A}\right\|$.

As a consequence of the above proposition, any instance of the A-BD-HPE framework of Section 2.1 with F, C and D as in (2.2.4) and the inner products $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$ defined as in (2.2.3) will satisfy the global convergence rate properties described in Theorems 2.7 and 2.8. Below we describe such an instance.

Algorithm 2.1 Scaled adaptive block-decomposition (SA-BD) method for (1.1.1).

0) Let $x^0 \in \mathcal{X}$, $y^0 \in \mathcal{Y}$, $0 < \sigma \leq 1$ and $\theta > 0$ be given, and set $k = 1$ and

$$\tilde{\lambda} = \frac{\sigma}{\|\mathcal{U}^{-1/2}\mathcal{A}\|}; \tag{2.2.5}$$

1) compute

$$\tilde{y}^k = y^{k-1} - \tilde{\lambda}\mathcal{U}^{-1}(\mathcal{A}x_{k-1} - b), \qquad \tilde{x}^k = \Pi_K\left[x^{k-1} - \tilde{\lambda}\left(c - \mathcal{A}^*\tilde{y}^k\right)\right]; \tag{2.2.6}$$

2) define

$$\tilde{v}_k = \begin{pmatrix} \mathcal{U}^{-1}(\mathcal{A}\tilde{x}^k - b) \\ (x^{k-1} - \tilde{x}^k)/\tilde{\lambda} \end{pmatrix}, \tag{2.2.7}$$

choose λ_k to be the largest $\lambda > 0$ such that

$$\left\|\lambda\tilde{v}_k + (\tilde{y}^k, \tilde{x}^k) - (y^{k-1}, x^{k-1})\right\|_{\mathcal{Z}\times\mathcal{W}} \leq \sigma\left\|(\tilde{y}^k, \tilde{x}^k) - (y^{k-1}, x^{k-1})\right\|_{\mathcal{Z}\times\mathcal{W}};$$

3) set $(y^k, x^k) = (y^{k-1}, x^{k-1}) - \lambda_k\tilde{v}_k$ and $k \leftarrow k + 1$, and go to step 1.

Proposition 2.10. *Let $\sigma_z = \tilde{\sigma}_z = \sigma_w = 0$ and define the inner products $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, and operators F, C and D according to* (2.2.3) *and* (2.2.4). *Consider the sequences*

$\{(x^k, y^k)\}$ and $\{(\tilde{x}^k, \tilde{y}^k)\}$ generated by Algorithm 2.1 and, for every $k \in \mathbb{N}$, define

$$z^k = y^k, \qquad w^k = x^k, \qquad \tilde{z}^k = \tilde{y}^k, \qquad \tilde{w}^k = \tilde{x}^k, \tag{2.2.8}$$

$$\tilde{\lambda}_k = \tilde{\lambda}, \quad \varepsilon_k^{\tilde{z}} = \varepsilon_k^w = 0, \quad \tilde{c}^k = 0, \quad \tilde{d}^k = -\tilde{s}^k, \tag{2.2.9}$$

where

$$\tilde{s}^k := c - \mathcal{A}^* \tilde{y}^k - \frac{1}{\tilde{\lambda}}(x^{k-1} - \tilde{x}^k). \tag{2.2.10}$$

Then the following statements hold for every $k \in \mathbb{N}$:

a) $\tilde{\lambda}_k$ satisfies (2.1.3) *with $L = \left\|\mathcal{U}^{-1/2}\mathcal{A}\right\|$;*

b) $\tilde{x}^k \in K$ and $\tilde{s}^k \in -N_K(\tilde{x}^k)$, or equivalently, $\tilde{x}^k \in K$, $\tilde{s}^k \in K^*$ and $\langle \tilde{x}^k, \tilde{s}^k \rangle = 0$;

c) $\tilde{\lambda}_k$, z^{k-1}, w^{k-1}, and the triples $(\tilde{z}^k, \tilde{c}^k, \varepsilon_k^{\tilde{z}})$ and $(\tilde{w}^k, \tilde{d}^k, \varepsilon_k^w)$ satisfy (2.1.4), (2.1.5) *and* (2.1.6).

As a consequence, Algorithm 2.1 is a special instance of the A-BD-HPE framework.

Proof. a) This statement follows straightforwardly from (2.2.5).

b) Define $w^k := x^{k-1} - \tilde{\lambda}\left(c - \mathcal{A}^*\tilde{y}^k\right)$ and note that $\tilde{x}^k = \Pi_K(w^k) \in K$ in view of (2.2.6). This together with (2.2.10) then imply that

$$\tilde{\lambda}\tilde{s}^k = \tilde{\lambda}(c - \mathcal{A}^*\tilde{y}^k) - (x^{k-1} - \Pi_K(w^k)) = -\left(w^k - \Pi_K(w^k)\right) \in -N_K\left(\tilde{x}^k\right), \tag{2.2.11}$$

where the inclusion follows from the well-known fact that $x - \Pi_K(x) \in N_K(\Pi_K(x))$ for all $x \in \mathcal{X}$. Statement b) now follows from the above observations and the fact that $N_K\left(\tilde{x}^k\right)$ is a cone. The equivalent statement of b) follows from Lemma 2.3 with $q = \tilde{s}^k$, $x = \tilde{x}^k$ and $\varepsilon = 0$.

c) It follows from (2.2.8), (2.2.9), (2.2.10), the definition of F in (2.2.4), and the definition of $\tilde{y}^k$ in (2.2.6) that

$$\tilde{\lambda}_k\left[F_1(\tilde{z}^k, w^{k-1}) + \tilde{c}^k\right] + \tilde{z}^k - z^{k-1} = 0, \qquad \tilde{\lambda}_k\left[F_2(\tilde{z}^k, \tilde{w}^k) + \tilde{d}^k\right] + \tilde{w}^k - w^{k-1} = 0.$$

Clearly, the fact that $\sigma_z = \tilde{\sigma}_z = \sigma_w = \varepsilon_k^{\tilde{z}} = \varepsilon_k^w = 0$, the two identities above and (2.2.8) imply that all the inequalities in (2.1.4), (2.1.5) and (2.1.6) are satisfied with . The inclusions

in (2.1.4) and (2.1.6) hold from the definitions of C, D, z^k, w^k, $\tilde{z}^k$, $\tilde{w}^k$, ε_k^z, ε_k^w, $\tilde{c}^k$ and $\tilde{d}^k$ in (2.2.4), (2.2.8) and (2.2.9), and the inclusion in (2.2.11). Hence, statement c) follows.

Finally, the definitions of F in (2.2.4), $\tilde{v}_k$ in (2.2.7), z^k, w^k, $\tilde{z}^k$ and $\tilde{w}^k$ in (2.2.8), and $\tilde{c}^k$ and $\tilde{d}^k$ in (2.2.9), imply that

$$\tilde{v}_k = F(\tilde{z}^k, \tilde{w}^k) + (\tilde{c}^k, \tilde{d}^k).$$

This observation together with the fact that $\varepsilon_k^z = \varepsilon_k^w = 0$ then imply that steps 2 and 3 of Algorithm 2.1 are equivalent to steps 3 and 4 of the A-BD-HPE framework. Therefore, the conclusion of the proposition follows. □

We now specialize the convergence rate results of Section 2.1, namely Theorems 2.7 and 2.8, to the context of Algorithm 2.1. First, define for every non-singular linear mapping $\mathcal{B} : \mathcal{Y} \to \mathcal{Y}$ and $y \in \mathcal{Y} \setminus \{0\}$ the following quantity

$$\xi(\mathcal{B}, y) := \frac{\|\mathcal{B}y\|}{\|\mathcal{B}\|\|y\|} \in \left[\frac{1}{\kappa(\mathcal{B})}, 1\right], \tag{2.2.12}$$

where $\kappa(\mathcal{B}) := \|\mathcal{B}^{-1}\|\|\mathcal{B}\|$ denotes the condition number of $\mathcal{B}$. Note that for any scalar $\theta \in \mathbb{R} \setminus \{0\}$ we have that $\xi(\theta\mathcal{B}, y) = \xi(\mathcal{B}, y)$ and $\xi(\theta\mathcal{I}, y) = 1$.

Theorem 2.11. *Consider the sequences $\{(x^k, y^k)\}$ and $\{(\tilde{x}^k, \tilde{y}^k)\}$ generated by Algorithm 2.1, the sequence $\{\tilde{s}^k\}$ defined as in* (2.2.10) *and, for every $k \in \mathbb{N}$, define*

$$(\tilde{x}^{a,k}, \tilde{y}^{a,k}, \tilde{s}^{a,k}) = \frac{1}{\Lambda_k} \sum_{i=1}^{k} \lambda_i (\tilde{x}^i, \tilde{y}^i, \tilde{s}^i). \tag{2.2.13}$$

Let X^* and Y^* denote the set of optimal solutions of (1.1.1) *and* (1.1.2)*, respectively, and $(x^*, y^*) \in X^* \times Y^*$ be such that*

$$d_{0,x} := \min\left\{\|x^0 - x\| : x \in X^*\right\} = \|x^0 - x^*\|, \quad d_{0,y} := \min\left\{\|y^0 - y\| : y \in Y^*\right\} = \|y^0 - y^*\|. \tag{2.2.14}$$

Then, for every $k \in \mathbb{N}$, the following statements hold:

a) $\tilde{x}^k \in K$, $\tilde{s}^k \in K^*$, $\langle \tilde{x}^k, \tilde{s}^k \rangle = 0$, and if $\sigma < 1$, there exists $i \leq k$ such that

$$\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|^2 + \|\mathcal{A}^*\tilde{y}^i + \tilde{s}^i - c\|^2 \leq \left(\frac{1+\sigma}{1-\sigma}\right) \|\mathcal{U}^{-1/2}\mathcal{A}\|^2 \frac{d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\| d_{0,y}^2}{k\sigma^2}, \tag{2.2.15}$$

where $\xi_0(\mathcal{U}) := [\xi(\mathcal{U}^{1/2}, y^0 - y^*)]^2$.

b) $\tilde{x}^{a,k} \in K$, $\tilde{s}^{a,k} \in K^*$ *and*

$$\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}^{a,k} - b)\|^2 + \|\mathcal{A}^*\tilde{y}^{a,k} + \tilde{s}^{a,k} - c\|^2 \leq \|\mathcal{U}^{-1/2}\mathcal{A}\|^2 \frac{d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\|d_{0,y}^2}{(k\sigma)^2},$$

$$\langle \tilde{x}^{a,k}, \tilde{s}^{a,k} \rangle \leq (2 + 8\sigma) \left\|\mathcal{U}^{-1/2}\mathcal{A}\right\| \frac{d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\|d_{0,y}^2}{k\sigma}.$$

Proof. We first prove statement a). Let $k \in \mathbb{N}$ be given. First note that from Proposition 2.10(d) we have $\tilde{s}^k \in K^*$ and $\langle \tilde{x}^k, \tilde{s}^k \rangle = 0$. Observe also that $X^* \times Y^*$ is the set of solutions of the inclusion problem (2.1.2) and (2.2.4) (see the fourth observation after (2.2.1)). Let d_0 denote the distance of (y^0, x^0) to $Y^* \times X^*$ with respect to the scaled norm $\|\cdot\|_{\mathcal{Z}\times\mathcal{W}}$, and observe that the definitions of (x^*, y^*), $d_{0,x}$, $d_{0,y}$ and $\xi_0(\mathcal{U})$ imply

$$d_0^2 \leq \left\|x^0 - x^*\right\|_{\mathcal{W}}^2 + \left\|y^0 - y^*\right\|_{\mathcal{Z}}^2 = \left\|x^0 - x^*\right\|^2 + \|\mathcal{U}^{1/2}(y^0 - y^*)\| = d_{0,x}^2 + \xi_0(\mathcal{U})\|\mathcal{U}\|d_{0,y}^2.$$

Moreover, by Proposition 2.10 and Theorem 2.7 with $\alpha = 1$, we conclude that if $\sigma < 1$, there exists $i \leq k$ such that

$$\begin{aligned} \|\mathcal{U}^{-1}(\mathcal{A}\tilde{x}^i - b) + \tilde{c}^i\|_{\mathcal{Z}}^2 + \|c - \mathcal{A}^*\tilde{y}^i + \tilde{d}^i\|_{\mathcal{W}}^2 &\leq \left(\tfrac{1+\sigma}{1-\sigma}\right) \frac{d_0^2}{\lambda_i \sum_{j=1}^k \lambda_j} \leq \left(\tfrac{1+\sigma}{1-\sigma}\right) \frac{d_0^2}{\tilde{\lambda}^2 k} \\ &= \left(\tfrac{1+\sigma}{1-\sigma}\right) \|\mathcal{U}^{-1/2}\mathcal{A}\|^2 \frac{d_0^2}{k\sigma^2}, \end{aligned}$$

where the second inequality follows from Proposition 2.6 with $\tilde{\lambda}_k = \tilde{\lambda}$, and the last equality follows from the definition of $\tilde{\lambda}$ in (2.2.5). Also, in view of (2.2.3) and the definitions of $\tilde{c}^i$ and $\tilde{d}^i$ in (2.2.9), we have

$$\|\mathcal{U}^{-1}(\mathcal{A}\tilde{x}^i - b) + \tilde{c}^i\|_{\mathcal{Z}}^2 = \|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|^2, \qquad \|c - \mathcal{A}^*\tilde{y}^i + \tilde{d}^i\|_{\mathcal{W}}^2 = \left\|\mathcal{A}^*\tilde{y}^i + \tilde{s}^i - c\right\|^2.$$

Then, combining the last four relations we obtain (2.2.15), and hence statement a) follows.

Statement b) can be proved in a similar way using Theorem 2.8 instead of Theorem 2.7. □

We now make several remarks about Theorem 2.11. For the sake of simplicity, we will focus our discussion on the point-wise convergence rate bound (2.2.15). Define the

self-adjoint positive definite linear mapping $\mathcal{U}_0 : \mathcal{Y} \to \mathcal{Y}$ as

$$\mathcal{U}_0 := \mathcal{A}\mathcal{A}^*.$$

First, the term $\|\mathcal{U}^{-1/2}\mathcal{A}\|$ in the right hand side of (2.2.15) is minimized over the class $\mathcal{C}(\mathcal{A})$ consisting of all self-adjoint positive definite linear mappings $\mathcal{U} : \mathcal{Y} \to \mathcal{Y}$ satisfying $\|\mathcal{U}^{-1/2}\mathcal{A}\|^2 = \|\mathcal{A}\|^2/\|\mathcal{U}\|$, or equivalently,

$$\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|^2 = \|\mathcal{U}_0\|/\|\mathcal{U}\|. \tag{2.2.16}$$

Note that any positive multiple of the identity operator $\mathcal{I}$ or the operator $\mathcal{U}_0$ belongs to $\mathcal{C}(\mathcal{A})$. In view of this remark, we will assume from now on that $\mathcal{U} \in \mathcal{C}(\mathcal{A})$, and within this class we will consider the subclass $\mathcal{C}_\theta(\mathcal{A})$ consisting of the operators $\mathcal{U} \in \mathcal{C}(\mathcal{A})$ such that $\|\mathcal{U}\| = \theta$, where $\theta > 0$ is some pre-specified scalar. Second, recall that the definition of the term $\xi_0(\mathcal{U})$ implies that $\xi_0(\mathcal{U}) \in [1/\kappa(\mathcal{U}), 1]$ and that $\xi_0(\theta\mathcal{I}) = 1$. Hence, $\mathcal{U} = \theta\mathcal{I}$ maximizes $\xi_0(\mathcal{U})$ over $\mathcal{C}_\theta(\mathcal{A})$. Also, it is interesting to observe that the best possible value $\xi_0(\mathcal{U})$ might take over $\mathcal{C}(\mathcal{A})$, namely $1/\kappa(\mathcal{U})$, achieves its minimum value when $\mathcal{U}$ is a positive multiple of $\mathcal{U}_0$. Indeed, in view of (2.2.16) and the definition of $\kappa(\cdot)$, we have

$$\begin{aligned}
\kappa(\mathcal{U}) &= \|\mathcal{U}\|\|\mathcal{U}^{-1}\| = \|\mathcal{U}\|\|\mathcal{U}^{-1/2}\|^2 \\
&\leq \|\mathcal{U}\|\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|^2\|\mathcal{U}_0^{-1/2}\|^2 = \|\mathcal{U}\|\frac{\|\mathcal{U}_0\|}{\|\mathcal{U}\|}\|\mathcal{U}_0^{-1}\| = \kappa(\mathcal{U}_0), \quad \forall \mathcal{U} \in \mathcal{C}(\mathcal{A}).
\end{aligned}$$

Third, if you view the vector $u^0 := (y^0 - y^*)/\|y^0 - y^*\|$ as being uniformly distributed on the unit sphere, then Lemma A.1 and the definition of $\xi_0(\mathcal{U})$ implies that the expected value of $\xi_0(\mathcal{U})$ with respect to u^0 is $\mathrm{tr}(\mathcal{U})/(m\|\mathcal{U}\|)$, or in words, the average of the eigenvalues of $\mathcal{U}$ divided by the maximum eigenvalue of $\mathcal{U}$. Hence, if $\mathcal{U}_0$ is such that $\mathrm{tr}(\mathcal{U}_0)/(m\|\mathcal{U}_0\|)$ is of the same order of magnitude as $1/\kappa(\mathcal{U}_0)$ and $\mathcal{U}_0$ is ill-conditioned, then the choice of $\mathcal{U} = \theta\mathcal{U}_0/\|\mathcal{U}_0\|$ from the class $\mathcal{C}_\theta(\mathcal{A})$ for Algorithm 2.1 will be nearly optimal in the sense of minimizing $\xi_0(\mathcal{U})$. Note that the latter condition happens when $\mathcal{U}_0$ is ill-conditioned and most of the eigenvalues of $\mathcal{U}_0$ are relatively close to its minimum eigenvalue.

We will now interpret the bound (2.2.15) from a geometrical point of view. Define the primal and dual manifolds as

$$\mathcal{M}_p := \{x : \mathcal{A}x = b\}, \qquad \mathcal{M}_d := \{c - \mathcal{A}^*y : y \in \mathcal{Y}\},$$

and define

$$\hat{\xi}_i(\mathcal{U}) \;:=\; \left(\frac{1}{\xi(\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}, \mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}^i - b))}\right)^2. \tag{2.2.17}$$

For every $\mathcal{U} \in \mathcal{C}_\theta(\mathcal{A})$, we can easily see that (2.2.15), (2.2.16), the definition of $\text{dist}_C(\cdot)$ in (1.4.1), and the fact that $\|\mathcal{U}\| = \theta$ and $\|\mathcal{U}^{-1/2}\mathcal{A}\| = \|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|$, imply

$$[\text{dist}_{\mathcal{M}_d}(\tilde{s}^i)]^2 \leq \|\mathcal{A}^*\tilde{y}^i + \tilde{s}^i - c\|^2 \leq \left(\frac{1+\sigma}{1-\sigma}\right)\frac{\|\mathcal{U}_0\|}{k\sigma^2}\left(\frac{d_{0,x}^2}{\theta} + \xi_0(\mathcal{U})d_{0,y}^2\right). \tag{2.2.18}$$

We will now bound the distance $\text{dist}_{\mathcal{M}_p}(\tilde{x}^i)$. First, it is easy to see that

$$\text{dist}_{\mathcal{M}_p}(\tilde{x}^i) = \|\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|.$$

Hence, for every $\mathcal{U} \in \mathcal{C}_\theta(\mathcal{A})$, we have that (2.2.15), (2.2.12), (2.2.17), and the fact that $\|\mathcal{U}\| = \theta$ and $\|\mathcal{U}^{-1/2}\mathcal{A}\| = \|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|$, imply

$$\begin{aligned}
[\text{dist}_{\mathcal{M}_p}(\tilde{x}^i)]^2 &= \|\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|^2 \\
&= \left(\frac{\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|}{\xi(\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}, \mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}^i - b))\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|}\right)^2 \\
&= \frac{\hat{\xi}_i(\mathcal{U})}{\|\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}\|^2}\|\mathcal{U}^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|^2 \\
&\leq \left(\frac{1+\sigma}{1-\sigma}\right)\frac{\hat{\xi}_i(\mathcal{U})}{k\sigma^2}\left(d_{0,x}^2 + \xi_0(\mathcal{U})\theta d_{0,y}^2\right).
\end{aligned} \tag{2.2.19}$$

Note that the definition of the term $\hat{\xi}_i(\mathcal{U})$ implies that $\hat{\xi}_i(\mathcal{U}) \in [1, (\kappa(\mathcal{U}^{-1/2}\mathcal{U}_0^{1/2}))^2]$ and that $\xi_0(\theta\mathcal{U}_0) = 1$. Hence, the choice of $\mathcal{U} = \theta\mathcal{U}_0/\|\mathcal{U}_0\|$ minimizes $\hat{\xi}_i(\mathcal{U})$ over $\mathcal{C}_\theta(\mathcal{A})$ which, in view of the observations about the term $\xi_0(\mathcal{U})$ above, can lead to nearly optimal bounds for the distances to the primal and dual manifolds in (2.2.19) and (2.2.18), respectively.

The convergence rate bounds in (2.2.18) and (2.2.19), not only highlight the benefits obtained by $\xi_0(\mathcal{U}) \leq 1$ for an ill-conditioned $\mathcal{U}$, but also suggest how the magnitude of $\|\mathcal{U}\| = \theta$ affects the size of the primal and dual residuals. More specifically, viewing all the quantities in (2.2.18) and (2.2.19), with the exception of θ, as constants, and noting that

$$[\text{dist}_{\mathcal{M}_p}(\tilde{x}^i)]^2 = \frac{\|\mathcal{U}_0^{-1/2}(\mathcal{A}\tilde{x}^i - b)\|^2\|\mathcal{U}_0^{1/2}\|^2}{\|\mathcal{A}\tilde{x}^i - b\|^2}\frac{\|\mathcal{A}\tilde{x}^i - b\|^2}{\|\mathcal{U}_0^{1/2}\|^2} = \frac{\hat{\xi}_i(\mathcal{I})\|\mathcal{A}\tilde{x}^i - b\|^2}{\|\mathcal{U}_0\|} \geq \frac{\|\mathcal{A}\tilde{x}^i - b\|^2}{\|\mathcal{U}_0\|},$$

we can see that the primal and dual residuals are

$$\|\mathcal{A}\tilde{x}^i - b\|^2 = \mathcal{O}\left(\max\left\{1, \theta^{1/2}\right\}\right), \qquad \|\mathcal{A}^*\tilde{y}^i + \tilde{s}^i - c\|^2 = \mathcal{O}\left(\max\left\{1, \theta^{-1/2}\right\}\right),$$

respectively. Hence, as $\theta \to 0$, the dual residual can become significantly larger than the primal one while, as $\theta \to \infty$, the primal residual can become significantly larger than the dual one. In fact, we have observed in our computational experiments that these residuals behave exactly as just described. In Subsection 2.2.2, we will use $\mathcal{U} = \theta\mathcal{U}_0$ and a dynamic choice of the scaling parameter θ in our implementation of Algorithm 2.1 so as to empirically balance the primal and dual residuals and as a consequence improve the practical performance of the method.

2.2.2 Implementation details and numerical results

In this subsection, we describe all the ingredients needed to speed-up the implementation of Algorithm 2.1, and present numerical results demonstrating the efficiency of the resulting method for solving many large instances of (1.1.1) and (1.1.4). More specifically, we describe two important ingredients, namely: i) convenient choice of initial primal and dual iterates, and initial parameter for the scaled inner product (2.2.3) on the space $\mathcal{Y}$, and; ii) dynamic change of the scaled inner product in the $\mathcal{Y}$ space as the algorithm progresses, with the aim of properly balancing the sizes of the primal and dual relative residuals. This subsection also contains five subsections reporting computational results which compare our method with the ones in [30, 52] and [68] for various types of conic semidefinite programming problems.

For every $k \in \mathbb{N}$, define the primal and dual relative residuals as

$$\epsilon_{P,k} := \frac{\|\mathcal{A}\tilde{x}^k - b\|}{1 + \|b\|}, \quad \epsilon_{D.k} := \frac{\|\mathcal{A}^*\tilde{y}^k + \tilde{s}^k - c\|}{1 + \|c\|}, \tag{2.2.20}$$

where $\{\tilde{x}^k\}$ and $\{\tilde{y}^k\}$ are the sequence generated by Algorithm 2.1, and $\{\tilde{s}^k\}$ is given by (2.2.10). In our implementation, we used the stopping criterion

$$\max\{\epsilon_{P,k}, \epsilon_{D,k}\} \leq \bar{\epsilon}, \tag{2.2.21}$$

where $\bar{\epsilon} > 0$ is a given tolerance. We observe that the complementarity measure is $\langle \tilde{x}^k, \tilde{s}^k \rangle = 0$ for every $k \in \mathbb{N}$, in view of Theorem 2.11(a). We note that the two methods we compare our code to also use the stopping criterion (2.2.21) and satisfy the later complementarity property.

Our implementation chooses the initial iterates x^0 and y^0 as

$$x^0 = 0, \qquad y^0 = \arg\min \|\mathcal{A}^* y - c\| = \mathcal{U}_0^{-1}\mathcal{A}c. \tag{2.2.22}$$

Another possibility would be to choose x^0 as the vector with minimum norm lying in the manifold $\{x \in \mathcal{X} : \mathcal{A}x = b\}$. However, the computational results reported in this section are based on the choice of the initial iterates given by (2.2.22).

Our benchmark is based on an implementation of Algorithm 2.1 in which $\sigma = 0.99$ and the operator $\mathcal{U}$ is chosen as

$$\mathcal{U} = \theta\mathcal{U}_0, \tag{2.2.23}$$

where θ is dynamically updated whenever a specified number of iterations is performed. In the next two paragraphs we discuss how to initialize θ and the scheme for dynamically updating it.

First we discuss how to initialize θ. Note that the choice (2.2.23) of $\mathcal{U}$ implies that $\left\|\mathcal{U}^{-1/2}\mathcal{A}\right\| = \theta^{-1/2}$, and hence

$$\tilde{\lambda} = \sigma\theta^{1/2},$$

in view of (2.2.5). This observation together with (2.2.20), (2.2.10) and (2.2.22) imply that the initial relative residuals $\epsilon_{P,1}$ and $\epsilon_{D,1}$ as a function of θ are given by

$$\epsilon_{P,1} = \epsilon_{P,1}(\theta) := \frac{\left\|\mathcal{A}\tilde{x}^1(\theta) - b\right\|}{1 + \|b\|}, \qquad \epsilon_{D,1} = \epsilon_{D,1}(\theta) := \frac{\|(x^0 - \tilde{x}^1(\theta))/\tilde{\lambda}\|}{1 + \|c\|} = \frac{\left\|\sigma^{-1}\theta^{-1/2}\tilde{x}^1(\theta)\right\|}{1 + \|c\|}, \tag{2.2.24}$$

where

$$\begin{aligned} \tilde{x}^1 = \tilde{x}^1(\theta) &:= \Pi_K\left[x^0 - \tilde{\lambda}\left(c - \mathcal{A}^*\tilde{y}^1\right)\right] = \Pi_K\left[x^0 - \tilde{\lambda}\left(c - \mathcal{A}^*(y^0 - \tilde{\lambda}\mathcal{U}^{-1}(\mathcal{A}x^0 - b))\right)\right] \\ &= \sigma\theta^{1/2}\Pi_K\left[-c + \mathcal{A}^*\left(y^0 + \sigma\theta^{1/2}\mathcal{U}^{-1}b\right)\right]. \end{aligned}$$

Using the definition of y^0 in (2.2.22), we easily see that $\left\|\mathcal{A}^* y^0 - c\right\| \le \|c\|$, and hence , as $\theta \to 0$, we have from (2.2.24) that

$$\epsilon_{P,1}(\theta) \to \frac{\|b\|}{1 + \|b\|} < 1, \qquad \epsilon_{D,1}(\theta) \to \frac{\left\|\Pi_K\left[\mathcal{A}^* y^0 - c\right]\right\|}{1 + \|c\|} \le \frac{\left\|\mathcal{A}^* y^0 - c\right\|}{1 + \|c\|} \le \frac{\|c\|}{1 + \|c\|} < 1,$$

As a consequence, we can always choose an initial θ so as to enforce $\max\{\epsilon_{P,1}, \epsilon_{D,1}\}$ to be $\mathcal{O}(1)$. In fact, in our implementation we use the following procedure. Given a constant

$\rho \geq 1$, we check whether $\max\{\epsilon_{P,1}(1), \epsilon_{D,1}(1)\} \leq \rho$. If so, we set the initial θ to be 1, otherwise we successively divide the current value of θ by 2 until $\max\{\epsilon_{P,1}(\theta), \epsilon_{D,1}(\theta)\} \leq \rho$ is satisfied, and use this value as the initial θ. The motivation behind this initial choice of θ is to guarantee that the initial primal and dual relative residuals $\epsilon_{P,k}$ and $\epsilon_{D,k}$ are not too large at the first iteration of Algorithm 2.1.

Even though, the convergence rate bounds of Theorem 2.11 are guaranteed for a fixed value of θ, we have used in our computational results the heuristic of changing θ every time a specified number $\bar{k}$ of iterations have been performed. The motivation for dynamically changing θ, is that our preliminary computational experiments have suggested us that the performance of the method is improved as $\epsilon_{P,k}$ and $\epsilon_{D,k}$ are of the same order of magnitude. More specifically, if θ_k denotes the dynamic value of θ at the kth iteration of the algorithm, we use the following rule for updating θ_k,

$$\theta_k = \begin{cases} \theta_{k-1}, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma^{-1} \leq \epsilon_{P,k-1}/\epsilon_{D,k-1} \leq \gamma \\ \theta_{k-1} \cdot \tau, & k \equiv 0 \bmod \bar{k} \text{ and } \epsilon_{P,k-1}/\epsilon_{D,k-1} > \gamma \\ \theta_{k-1}/\tau, & k \equiv 0 \bmod \bar{k} \text{ and } \epsilon_{D,k-1}/\epsilon_{P,k-1} > \gamma \end{cases}, \qquad \forall k \geq 2,$$

for some pre-specified integer $\bar{k} \geq 1$, and scalars $\gamma > 1$ and $0 < \tau < 1$. In our computational experiments, we have used $\bar{k} = 5$, $\gamma = 1.5$ and $\tau = 0.9$. Note that this update rule is motivated by the last observation in Section 2.2.1. In summary, the update rule changes the value of θ at most a single time in the right direction, so as to balance the sizes of the primal and dual relative residuals based on the information provided by their values at the previous iteration. We should emphasize that convergence rate bounds for Algorithm 2.1 endowed with this updating rule are not available, but we have observed that this variant of Algorithm 2.1 performs extremely well.

In our computational experiments, we will refer to the variant of Algorithm 2.1 described above as the *dynamically scaled adaptive block-decomposition* (DSA-BD) method for solving (1.1.1). This variant was implemented for spaces $\mathcal{X}$ and $\mathcal{Y}$, and cone K given as in (1.1.4). Hence, our code is able to solve conic programming problems given in standard form (i.e., as in (1.1.1)) with n_u unrestricted scalar variables, n_l nonnegative scalar variables and

an $n_s \times n_s$ positive semidefinite symmetric matrix variable. The inner products (before scaling) used in $\mathcal{X}$ and $\mathcal{Y}$ are the standard ones, namely: the scalar inner product in $\mathcal{Y}$ and the following inner product in $\mathcal{X}$

$$\langle x, \tilde{x} \rangle := x_v^T \tilde{x}_v + X_s \bullet \tilde{X}_s,$$

for every $x = (x_v, X_s) \in \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}$ and $\tilde{x} = (\tilde{x}_v, \tilde{X}_s) \in \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}$, where $X \bullet Y := Tr(X^T Y)$ for every $X, Y \in \mathcal{S}^{n_s}$.

We present a computational benchmark of our algorithm (DSA-BD) compared to the semismooth Newton-CG augmented Lagrangian (SDPNAL) method in [68] and the boundary-point (BP) method in [30, 52]. We implemented the DSA-BD method in MATLAB using the SDPT3 data structures described in [64], but without exploiting any possible block sparsity on the semidefinite variable X_s. All the tests were made using a server with 2 Xeon X5460 processors at 3.16GHz and 32GB RAM.

Various large-scale SDP problems are solved to obtain this benchmark, ranging from purely random to SDP relaxations of combinatorial optimization problems such as the frequency assignment problem (FAP), the binary integer quadratic (BIQ) problem, the quadratic assignment problem (QAP) and the maximum stable set problem (Lovász θ-function and θ_+-function). In the following subsections, we describe in detail the problems included in our computational tests but before that, we make some general remarks about how the results are reported on the several tables given below. In Tables 2.2.1 and 2.2.11, we compare our method to BP and SDPNAL methods while, in Tables 2.2.3, 2.2.5, 2.2.6, 2.2.9 and 2.2.13, we compare it against SDPNAL only due to the fact that the current version of the BP method available to us only accepts conic optimization SDP problems without nonnegative scalar variables. In some of these tables, we report computational results for the same problem using two different tolerances. They are listed in two different rows of the table to the right of the name and size of the instance. We mark the time and the residual for a method in red, and also with an asterisk (*), whenever the instance cannot be solved to the required accuracy, with the convention that the time and residual reported are the ones obtained at the last iteration of the method. Also, the time marked in blue in a row is

the best one among the times listed in that row under the convention that when a method cannot solve the instance, the corresponding time is assumed to be ∞.

Observe that the final relative residuals obtained by BP and DSA-BD are very close to the desired accuracy when the latter is achieved. On the other hand, the ones obtained by SDPNAL can be noticeably smaller than the desired accuracy when the latter is achieved. This is due to the fact that SDPNAL is a second-order method and therefore it performs much fewer (and computationally more expensive) iterations than the other two methods. As a result, SDPNAL improves the relative residuals in a single iteration substantially more than the other two methods.

In Tables 2.2.2, 2.2.4, 2.2.7, 2.2.8, 2.2.10, 2.2.12 and 2.2.14, we report more detailed computational results obtained by our method DSA-BD. We do not report the violations to the conditions $\tilde{x} \in K$, $\tilde{s} \in K^*$, $\langle \tilde{x}, \tilde{s} \rangle = 0$, since they are satisfied up to machine precision at every iteration of the DSA-BD algorithm applied to all the instances in our benchmark.

Finally, we recall the following definition of a performance profile. For a given instance, a method A is said to be at most x times slower than method B, if the time taken by method A is at most x times the time taken by method B. A point (x, y) is in the performance profile curve of a method if it can solve exactly $(100y)\%$ of all the tested instances x times slower than any other competing method. Figure 2.2.1 plots the performance profiles (see [15]) of DSA-BD and SDPNAL methods based on all instances used in our benchmark. Note that the curve for SDPNAL becomes flat for $x \geq 6$ at a y value equal to about 0.5. This is due to the fact that SDPNAL fails to solve about 50% of the instances, although it is faster than DSA-BD on 18% of the instances. Other performance profiles based on instances belonging to the same class of conic programming problems will be reported in the subsequent subsections.

2.2.2.1 Random SDPs

This subsection compares the performance of our method DSA-BD with that of BP and SDPNAL on a collection of random sparse SDP problems. These instances were also used in [30] to report the performance of BP introduced in [52].

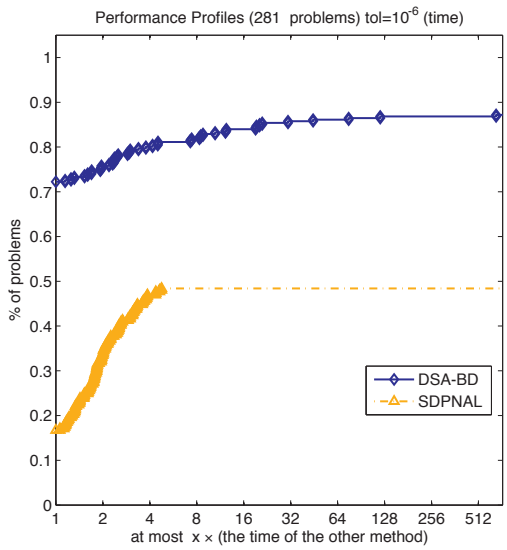

Figure 2.2.1: Performance profiles of DSA-BD and SDPNAL for solving 281 conic SDP problems with accuracy $\bar{\epsilon} = 10^{-6}$.

Table 2.2.1 compares the three methods on a collection of random sparse SDP instances using the tolerance $\bar{\epsilon} = 10^{-6}$. Table 2.2.2 gives more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figure 2.2.2 plots the performance profiles of the three methods based on these random sparse SDP instances only.

Note that DSA-BD finds a solution with an accuracy of at least 10^{-6} faster than BP and SDPNAL do in most of the random sparse SDP instances tested. In particular, DSA-BD is the fastest method on the larger instances.

2.2.2.2 Frequency assignment problems

This subsection compares the performance of our method DSA-BD with that of SDPNAL on a collection of SDP relaxations of FAPs.

The SDP relaxation of the FAP can be described as follows (see for example Subsection 2.4 in [9]). Given a network represented by a graph G with n nodes and an edge-weight

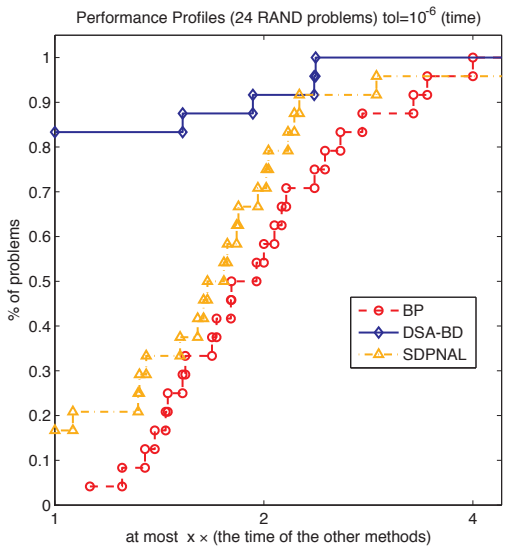

Figure 2.2.2: Performance profiles of DSA-BD, SDPNAL and BP for solving 24 random SDP problems with accuracy $\bar{\epsilon} = 10^{-6}$.

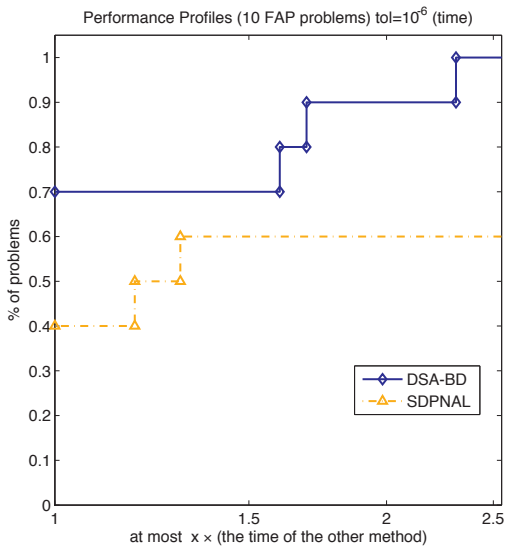

Figure 2.2.3: Performance profiles of DSA-BD and SDPNAL for solving 10 SDP relaxations of FAPs with accuracy $\bar{\epsilon} = 10^{-6}$.

matrix W, the frequency assignment problem on G can be formulated as a κ-cut problem

$$\begin{aligned} \max_{X \in \mathcal{S}^n} \quad & \left[\left(\frac{\kappa - 1}{2\kappa} \right) L(G, W) - \frac{1}{2} \mathrm{Diag}(We) \right] \bullet X \\ \text{s.t.} \quad & -E^{ij} \bullet X \leq 2/(\kappa - 1), \quad \forall\ (i,j), \\ & -E^{ij} \bullet X = 2/(\kappa - 1), \quad \forall\ (i,j) \in U \subseteq E, \\ & \mathrm{diag}(X) = e,\ X \succeq 0,\ \mathrm{rank}(X) = \kappa, \end{aligned}$$

where $\kappa > 1$ is an integer, $L(G,W) := \mathrm{Diag}(We) - W$ is the Laplacian matrix, $E^{ij} = e_i e_j^T + e_j e_i^T$ with $e_i \in \mathbb{R}^n$ the vector with all zeros except in the ith position and $e \in \mathbb{R}^n$ is the vector with all ones. An SDP relaxation of the problem above is obtained by dropping the rank restriction and the inequality constraint for the non-edges to obtain the following formulation

Table 2.2.1: Comparison of the methods on random SDP problems

Problem		$\max\{\epsilon_P, \epsilon_D\}$			Time		
Instance	$n_s \vert m$	BP	DSA-BD	SDPNAL	BP	DSA-BD	SDPNAL
RAND-0.3k10k	300—10000	1.0 -6	9.4 -7	8.5 -7	29	27	14
RAND-0.4k15k	400—15000	1.0 -6	9.8 -7	8.0 -7	61	52	22
RAND-0.6k20k	600—20000	9.8 -7	1.0 -6	5.5 -7	144	55	36
RAND-0.5k20k	500—20000	9.8 -7	9.7 -7	7.5 -7	110	76	32
RAND-0.3k20k	300—20000	9.5 -7	9.8 -7	6.6 -7	26	18	39
RAND-0.3k25k	300—25000	9.8 -7	1.0 -6	6.8 -7	73	65	69
RAND-0.4k30k	400—30000	9.4 -7	9.7 -7	8.4 -7	37	24	54
RAND-0.5k30k	500—30000	9.5 -7	9.6 -7	7.1 -7	52	29	53
RAND-0.4k40k	400—40000	9.2 -7	9.4 -7	8.6 -6*	115	92	134*
RAND-0.5k40k	500—40000	1.0 -6	9.6 -7	9.4 -7	53	31	57
RAND-0.6k40k	600—40000	9.5 -7	9.8 -7	7.2 -7	87	41	68
RAND-0.7k50k	700—50000	9.8 -7	9.7 -7	7.7 -7	140	65	88
RAND-0.6k50k	600—50000	9.6 -7	9.6 -7	6.1 -7	82	42	122
RAND-0.5k50k	500—50000	9.8 -7	9.6 -7	9.5 -7	89	66	100
RAND-0.6k60k	600—60000	9.5 -7	9.2 -7	9.2 -7	96	57	101
RAND-0.8k70k	800—70000	1.0 -6	9.8 -7	7.2 -7	196	80	131
RAND-0.7k70k	700—70000	9.6 -7	9.9 -7	6.3 -7	126	63	127
RAND-0.7k90k	700—90000	9.8 -7	9.5 -7	8.3 -7	202	145	192
RAND-0.9k100k	900—100000	1.0 -6	1.0 -6	6.3 -7	263	102	200
RAND-0.8k100k	800—100000	9.8 -7	9.6 -7	6.8 -7	203	113	250
RAND-1.0k100k	1000—100000	9.7 -7	9.7 -7	7.1 -7	450	137	240
RAND-0.8k110k	800—110000	9.9 -7	9.8 -7	3.8 -7	252	165	265
RAND-0.9k140k	900—140000	9.3 -7	9.6 -7	9.2 -7	384	264	348
RAND-1.0k150k	1000—150000	9.9 -7	9.7 -7	8.4 -7	459	194	394

Table 2.2.2: DSA-BD results on random SDP problems

Instance	$n_s \vert m$	$\langle C, X \rangle$	$b^T y$	Iter	ϵ_P	ϵ_D	Time
RAND-0.3k10k	300—10000	1.6597390 +2	1.6597440 +2	185	9.4 -7	4.0 -9	27
RAND-0.4k15k	400—15000	-6.5500010 +2	-6.5500030 +2	202	9.8 -7	5.1 -9	52
RAND-0.6k20k	600—20000	1.0452686 +3	1.0452662 +3	194	8.6 -7	1.0 -6	55
RAND-0.5k20k	500—20000	3.2800420 +2	3.2800230 +2	206	8.6 -7	9.7 -7	76
RAND-0.3k20k	300—20000	7.6135160 +2	7.6135210 +2	170	9.8 -7	9.3 -9	18
RAND-0.3k25k	300—25000	7.3838100 +1	7.3838400 +1	264	1.0 -6	5.2 -9	65
RAND-0.4k30k	400—30000	1.0721414 +3	1.0721394 +3	168	9.7 -7	8.9 -9	24
RAND-0.5k30k	500—30000	1.1076268 +3	1.1076267 +3	215	9.6 -7	5.5 -9	29
RAND-0.4k40k	400—40000	8.0576960 +2	8.0576860 +2	196	9.4 -7	6.0 -9	92
RAND-0.5k40k	500—40000	8.1661180 +2	8.1661080 +2	183	9.6 -7	5.0 -9	31
RAND-0.6k40k	600—40000	3.0661780 +2	3.0661720 +2	209	9.8 -7	1.0 -8	41
RAND-0.7k50k	700—50000	3.1320370 +2	3.1320280 +2	236	9.7 -7	4.7 -9	65
RAND-0.6k50k	600—50000	-3.8641380 +2	-3.8641360 +2	196	9.6 -7	8.2 -9	42
RAND-0.5k50k	500—50000	3.6494490 +2	3.6494520 +2	177	9.6 -7	4.9 -9	66
RAND-0.6k60k	600—60000	6.4173730 +2	6.4173680 +2	183	9.2 -7	5.6 -9	57
RAND-0.8k70k	800—70000	2.3313969 +3	2.3313957 +3	220	9.8 -7	9.4 -9	80
RAND-0.7k70k	700—70000	-3.6955780 +2	-3.6955870 +2	190	9.9 -7	6.6 -9	63
RAND-0.7k90k	700—90000	-2.6758200 +1	-2.6755500 +1	183	9.5 -7	7.9 -9	145
RAND-0.9k100k	900—100000	9.5422350 +2	9.5422290 +2	206	1.0 -6	7.9 -9	102
RAND-0.8k100k	800—100000	2.2592888 +3	2.2592881 +3	196	9.6 -7	5.8 -9	113
RAND-1.0k100k	1000—100000	3.0963606 +3	3.0963602 +3	235	9.7 -7	8.9 -9	137
RAND-0.8k110k	800—110000	1.8579204 +3	1.8579207 +3	187	9.8 -7	9.3 -9	165
RAND-0.9k140k	900—140000	2.3198295 +3	2.3198295 +3	193	9.6 -7	9.4 -9	264
RAND-1.0k150k	1000—150000	1.0528840 +3	1.0528864 +3	205	9.7 -7	4.9 -9	194

$$
\begin{aligned}
\max_{X \in \mathcal{S}^n} \quad & \left[\left(\frac{\kappa - 1}{2\kappa} \right) L(G, W) - \frac{1}{2} \mathrm{Diag}(We) \right] \bullet X \\
\text{s.t.} \quad & X \succeq 0, && \text{(2.2.25a)} \\
& -E^{ij} \bullet X \leq 2/(\kappa - 1) \quad \forall\, (i,j) \in E \setminus U, && \text{(2.2.25b)} \\
& -E^{ij} \bullet X = 2/(\kappa - 1) \quad \forall\, (i,j) \in U \subseteq E,\ \mathrm{diag}(X) = e. && \text{(2.2.25c)}
\end{aligned}
$$

Table 2.2.3 compares the two methods on a collection of SDP relaxations of FAPs using the tolerances $\bar{\epsilon} = 10^{-5}, 10^{-6}$. In this table, computational results for each instance are reported in two rows, the first one for $\bar{\epsilon} = 10^{-5}$, and the second one for $\bar{\epsilon} = 10^{-6}$. Table 2.2.4 gives more detailed computational results on these instances obtained by our method

Table 2.2.3: Comparison of the methods on FAPs

Instance	Problem $n_s; n_l \vert m$	$\max\{\epsilon_P, \epsilon_D\}$ DSA-BD	$\max\{\epsilon_P, \epsilon_D\}$ SDPNAL	Time DSA-BD	Time SDPNAL
fap05	84;3263—3570	9.5 -6	1.7 -5*	8	18*
		9.9 -7	1.7 -5*	14	18*
fap06	93;3997—4371	9.9 -6	9.4 -6	8	10
		1.0 -6	6.7 -7	11	13
fap07	98;4139—4851	1.0 -5	9.4 -6	5	8
		1.0 -6	9.9 -7	10	13
fap08	120;6668—7260	1.0 -5	7.6 -6	9	6
		9.9 -7	9.5 -7	16	10
fap09	174;14025—15225	9.7 -6	8.8 -6	15	15
		9.9 -7	9.4 -7	19	19
fap10	183;13754—14479	9.9 -6	6.8 -6	38	17
		1.0 -6	9.3 -7	74	32
fap11	252;23275—24292	9.9 -6	6.9 -6	65	39
		1.0 -6	9.7 -7	132	78
fap12	369;24410—26462	1.0 -5	8.2 -6	103	79
		1.0 -6	2.5 -6*	259	188*
fap25	2118;311044—322924	1.0 -5	7.4 -6	7444	14517
		9.9 -7	5.1 -6*	23572	29557*
fap36	4110;1112293—1154467	1.0 -5	8.5 -6	49617	29485
		9.8 -7	5.6 -6*	148855	72063*

Table 2.2.4: DSA-BD results on FAPs

Instance	$n_s; n_l \vert m$	$\langle C, X \rangle$	$b^T y$	Iter	ϵ_P	ϵ_D	Time
fap05	84;3263—3570	3.0830000 -1	3.0830000 -1	2424	9.9 -7	7.8 -7	14
fap06	93;3997—4371	4.5933000 -1	4.5934000 -1	1694	1.0 -6	5.5 -7	11
fap07	98;4139;—4851	2.1176000 +0	2.1176000 +0	1766	1.0 -6	4.2 -7	10
fap08	120;6668—7260	2.4363000 +0	2.4363000 +0	1735	9.9 -7	8.2 -7	16
fap09	174;14025;—15225	1.0797800 +1	1.0797800 +1	1128	9.9 -7	8.6 -7	19
fap10	183;13754;—14479	9.6383000 -3	9.7289000 -3	3315	1.0 -6	8.0 -7	74
fap11	252;23275—24292	2.9713000 -2	2.9856000 -2	3380	1.0 -6	8.0 -7	132
fap12	369;24410—26462	2.7317000 -1	2.7341000 -1	4004	1.0 -6	8.2 -7	259
fap25	2118;311044—322924	1.2877800 +1	1.2880000 +1	5576	9.9 -7	8.1 -7	23572
fap36	4110;1112293—1154467	6.9857000 +1	6.9860700 +1	4013	9.8 -7	9.4 -7	148855

DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figure 2.2.3 plots the performance profiles of both methods based on these SDP relaxations of FAPs.

Note that our method performs better than SDPNAL on large SDP relaxations of FAPs (i.e., `fap25` and `fap36`).

2.2.2.3 Binary integer quadratic problems

This subsection compares the performance of our method DSA-BD with that of SDPNAL on a collection of SDP relaxations of BIQ problems.

The SDP relaxation of the BIQ problem can be described as follows (see for example Section 7 in [68]). Given an $n \times n$ symmetric matrix Q, the BIQ problem can be formulated as

$$\min \left\{ z^T Q z : z \in \{0,1\}^n \right\}.$$

By representing the binary set $\{0,1\}^n$ as $\left\{ z \in \mathbb{R}^n | z_i^2 - z_i = 0 \right\}$, we obtain the following SDP relaxation

$$
\begin{aligned}
\min \quad & Q \bullet Z \\
\text{s.t.} \quad & x := \begin{bmatrix} Z & z \\ z^T & \alpha \end{bmatrix} \succeq 0, && \text{(2.2.26a)} \\
& \operatorname{diag}(Z) - z = 0,\ \alpha = 1,\ Z \geq 0,\ z \geq 0, && \text{(2.2.26b)}
\end{aligned}
$$

where $Z \in \mathcal{S}^n$, $z \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$.

Tables 2.2.5 and 2.2.6 compare the two methods on a collection of SDP relaxations of BIQ problems using the tolerance $\bar{\epsilon} = 10^{-6}$. Tables 2.2.7 and 2.2.8 give more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figure 2.2.4 plots the performance profiles of both methods based on these SDP relaxations of BIQ problems.

Note that SDPNAL takes more time than DSA-BD to find a solution with an accuracy of at least 10^{-6} in almost all of the SDP relaxations of BIQ problems tested, and it fails to compute such a solution on more than half of these instances. On the other hand, our method DSA-BD was able to find a solution with an accuracy of at least 10^{-6} for all of the SDP relaxations of BIQ problems tested.

2.2.2.4 Quadratic assignment problems

This subsection compares the performance of our method DSA-BD with that of SDPNAL on a collection of SDP relaxations of QAPs.

Given the set Π of $n \times n$ permutation matrices and $A, B \in \mathbb{R}^{n \times n}$, the quadratic assignment problem can be formulated as

$$
\min \left\{ \langle X, AXB \rangle : X \in \Pi \right\}.
$$

For a matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{n \times n}$, we will identify it with the n^2-vector $x = (x_1; \ldots; x_n)$. For a matrix $Y \in \mathbb{R}^{n^2 \times n^2}$, we let Y^{ij} be the $n \times n$ block corresponding to $x_i x_j^T$ in the matrix

Table 2.2.5: Comparison of the methods on BIQ problems

Problem		$\max\{\epsilon_P, \epsilon_D\}$		Time	
Instance	$n_s; n_l \| m$	DSA-BD	SDPNAL	DSA-BD	SDPNAL
be100.1	101;5151—5252	1.0 -6	7.3 -7	21	55
be100.10	101;5151—5252	1.0 -6	8.3 -7	13	41
be100.2	101;5151—5252	1.0 -6	8.3 -7	17	51
be100.3	101;5151—5252	1.0 -6	9.0 -7	18	69
be100.4	101;5151—5252	1.0 -6	6.7 -6*	19	83*
be100.5	101;5151—5252	1.0 -6	6.8 -7	17	51
be100.6	101;5151—5252	9.9 -7	8.2 -6*	14	68*
be100.7	101;5151—5252	1.0 -6	8.0 -7	17	30
be100.8	101;5151—5252	9.2 -7	8.2 -7	13	21
be100.9	101;5151—5252	1.0 -6	5.0 -7	16	58
be120.3.1	121;7381—7502	9.9 -7	7.5 -7	25	80
be120.3.10	121;7381—7502	9.9 -7	6.6 -7	21	100
be120.3.2	121;7381—7502	1.0 -6	7.7 -6*	25	88*
be120.3.3	121;7381—7502	1.0 -6	6.0 -7	19	36
be120.3.4	121;7381—7502	9.8 -7	8.3 -7	20	51
be120.3.5	121;7381—7502	1.0 -6	1.3 -5*	30	98*
be120.3.6	121;7381—7502	1.0 -6	1.5 -5*	29	93*
be120.3.7	121;7381—7502	1.0 -6	4.1 -5*	46	102*
be120.3.8	121;7381—7502	1.0 -6	3.5 -5*	40	104*
be120.3.9	121;7381—7502	1.0 -6	5.5 -5*	38	74*
be120.8.1	121;7381—7502	9.9 -7	5.9 -7	21	52
be120.8.10	121;7381—7502	1.0 -6	9.9 -7	26	65
be120.8.2	121;7381—7502	1.0 -6	2.9 -5*	32	105*
be120.8.3	121;7381—7502	1.0 -6	7.6 -7	27	49
be120.8.4	121;7381—7502	1.0 -6	9.5 -6*	25	82*
be120.8.5	121;7381—7502	1.0 -6	8.9 -6*	27	111*
be120.8.6	121;7381—7502	9.9 -7	9.8 -7	24	91
be120.8.7	121;7381—7502	9.9 -7	6.6 -7	20	45
be120.8.8	121;7381—7502	9.7 -7	8.5 -7	19	19
be120.8.9	121;7381—7502	1.0 -6	9.9 -7	19	34
be150.3.1	151;11476—11627	1.0 -6	1.5 -5*	35	146*
be150.3.10	151;11476—11627	1.0 -6	1.3 -5*	53	177*
be150.3.2	151;11476—11627	1.0 -6	2.0 -5*	45	125*
be150.3.3	151;11476—11627	1.0 -6	6.1 -7	36	133
be150.3.4	151;11476—11627	1.0 -6	6.7 -7	36	60
be150.3.5	151;11476—11627	1.0 -6	9.0 -6*	44	135*
be150.3.7	151;11476—11627	1.0 -6	5.2 -7	36	94
be150.3.8	151;11476—11627	1.0 -6	2.2 -5*	53	84*
be150.3.9	151;11476—11627	9.5 -7	7.1 -7	35	49
be150.8.1	151;11476—11627	1.0 -6	8.2 -7	32	140
be150.8.10	151;11476—11627	1.0 -6	6.2 -7	38	80
be150.8.2	151;11476—11627	1.0 -6	6.9 -7	34	63
be150.8.3	151;11476—11627	1.0 -6	1.0 -6	36	119
be150.8.4	151;11476—11627	1.0 -6	5.7 -7	40	78
be150.8.5	151;11476—11627	9.9 -7	1.1 -5*	35	146*
be150.8.6	151;11476—11627	1.0 -6	9.4 -6*	40	119*
be150.8.7	151;11476—11627	1.0 -6	2.3 -5*	45	143*
be150.8.8	151;11476—11627	9.8 -7	1.3 -5*	52	136*
be150.8.9	151;11476—11627	1.0 -6	8.4 -7	46	108
be200.3.1	201;20301—20502	1.0 -6	4.8 -6*	67	204*
be200.3.10	201;20301—20502	1.0 -6	1.9 -5*	92	242*
be200.3.2	201;20301—20502	1.0 -6	6.7 -7	73	134
be200.3.3	201;20301—20502	1.0 -6	3.6 -5*	138	271*
be200.3.4	201;20301—20502	1.0 -6	1.6 -5*	87	258*
be200.3.5	201;20301—20502	1.0 -6	1.4 -5*	111	266*
be200.3.6	201;20301—20502	1.0 -6	7.2 -7	68	182
be200.3.7	201;20301—20502	1.0 -6	6.8 -7	84	261
be200.3.8	201;20301—20502	1.0 -6	9.1 -7	76	158
be200.3.9	201;20301—20502	1.0 -6	3.2 -5*	161	213*
be200.8.1	201;20301—20502	1.0 -6	3.0 -5*	117	234*
be200.8.10	201;20301—20502	1.0 -6	1.6 -5*	77	243*
be200.8.2	201;20301—20502	1.0 -6	5.3 -7	61	74
be200.8.3	201;20301—20502	1.0 -6	9.5 -6*	94	261*
be200.8.4	201;20301—20502	1.0 -6	5.8 -7	66	133
be200.8.5	201;20301—20502	1.0 -6	1.2 -5*	81	255*
be200.8.6	201;20301—20502	1.0 -6	1.7 -5*	94	269*
be200.8.7	201;20301—20502	1.0 -6	7.0 -7	69	119
be200.8.8	201;20301—20502	1.0 -6	6.6 -7	78	153
be200.8.9	201;20301—20502	1.0 -6	1.2 -5*	87	246*
be250.1	251;31626—31877	1.0 -6	1.0 -4*	209	360*
be250.10	251;31626—31877	1.0 -6	4.8 -5*	230	387*
be250.2	251;31626—31877	1.0 -6	3.0 -5*	183	450*
be250.3	251;31626—31877	9.9 -7	4.8 -5*	161	420*
be250.4	251;31626—31877	1.0 -6	6.5 -5*	362	420*
be250.5	251;31626—31877	1.0 -6	3.3 -5*	193	358*
be250.6	251;31626—31877	1.0 -6	4.2 -5*	188	400*
be250.7	251;31626—31877	1.0 -6	3.6 -5*	207	367*
be250.8	251;31626—31877	1.0 -6	1.6 -5*	176	444*
be250.9	251;31626—31877	1.0 -6	1.4 -4*	239	359*
bqp100-1	101;5151—5252	9.9 -7	7.6 -7	15	58
bqp100-10	101;5151—5252	1.0 -6	1.7 -5*	27	72*
bqp100-2	101;5151—5252	1.0 -6	1.4 -4*	28	60*
bqp100-3	101;5151—5252	1.0 -6	8.0 -7	43	65
bqp100-4	101;5151—5252	1.0 -6	7.7 -6*	25	84*
bqp100-5	101;5151—5252	1.0 -6	4.0 -5*	26	64*
bqp100-6	101;5151—5252	1.0 -6	8.6 -7	16	74
bqp100-7	101;5151—5252	1.0 -6	8.7 -7	14	61
bqp100-8	101;5151—5252	1.0 -6	2.1 -5*	25	75*

Table 2.2.6: Comparison of the methods on BIQ problems

Problem		$\max\{\epsilon_P, \epsilon_D\}$		Time	
Instance	$n_s; n_l \vert m$	DSA-BD	SDPNAL	DSA-BD	SDPNAL
bqp100-9	101;5151—5252	1.0 -6	3.7 -5*	35	76*
bqp250-1	251;31626—31877	1.0 -6	5.2 -7	182	391
bqp250-10	251;31626—31877	1.0 -6	9.5 -7	131	469
bqp250-2	251;31626—31877	1.0 -6	6.1 -5*	191	404*
bqp250-3	251;31626—31877	1.0 -6	8.0 -7	209	266
bqp250-4	251;31626—31877	1.0 -6	2.2 -5*	151	398*
bqp250-5	251;31626—31877	1.0 -6	5.2 -5*	272	418*
bqp250-6	251;31626—31877	1.0 -6	4.0 -5*	221	421*
bqp250-7	251;31626—31877	1.0 -6	5.5 -7	193	400
bqp250-8	251;31626—31877	1.0 -6	9.4 -6*	144	392*
bqp250-9	251;31626—31877	1.0 -6	1.7 -5*	195	396*
bqp50-1	51;1326—1377	1.0 -6	1.2 -5*	9	32*
bqp50-10	51;1326—1377	9.5 -7	5.8 -7	6	8
bqp50-2	51;1326—1377	9.9 -7	2.5 -5*	20	19*
bqp50-3	51;1326—1377	1.0 -6	8.2 -7	7	23
bqp50-4	51;1326—1377	1.0 -6	4.9 -5*	42	37*
bqp50-5	51;1326—1377	1.0 -6	4.4 -5*	9	19*
bqp50-7	51;1326—1377	1.0 -6	4.9 -7	7	8
bqp50-8	51;1326—1377	9.6 -7	6.9 -7	7	14
bqp50-9	51;1326—1377	1.0 -6	6.2 -7	6	8
bqp500-1	501;125751—126252	1.0 -6	4.2 -7	1230	1385
bqp500-10	501;125751—126252	1.0 -6	9.7 -7	1241	1893
bqp500-2	501;125751—126252	1.0 -6	6.2 -5*	1364	2113*
bqp500-3	501;125751—126252	1.0 -6	9.2 -7	1232	1433
bqp500-4	501;125751—126252	1.0 -6	9.6 -7	1303	1852
bqp500-5	501;125751—126252	1.0 -6	6.6 -5*	1294	2210*
bqp500-6	501;125751—126252	1.0 -6	9.3 -7	1236	2198
bqp500-7	501;125751—126252	1.0 -6	5.2 -5*	1280	1916*
bqp500-8	501;125751—126252	1.0 -6	7.7 -7	1314	1871
bqp500-9	501;125751—126252	1.0 -6	5.7 -5*	1187	2195*
gka10b	126;8001—8127	1.0 -6	2.3 -4*	22	68*
gka10d	101;5151—5252	1.0 -6	5.8 -7	17	31
gka1b	21;231—252	9.2 -7	2.9 -7	3	1
gka1c	41;861—902	1.0 -6	1.4 -5*	11	21*
gka1d	101;5151—5252	1.0 -6	1.6 -5*	33	68*
gka1e	201;20301—20502	1.0 -6	7.8 -5*	181	364*
gka1f	501;125751—126252	1.0 -6	7.4 -5*	2195	3531*
gka2b	31;496—527	1.0 -6	7.9 -5*	6	13*
gka2c	51;1326—1377	9.9 -7	5.4 -7	7	17
gka2d	101;5151—5252	1.0 -6	7.7 -6*	17	76*
gka2e	201;20301—20502	1.0 -6	9.2 -7	121	239
gka2f	501;125751—126252	1.0 -6	7.9 -5*	2488	3729*
gka3a	71;2556—2627	9.9 -7	7.9 -7	11	28
gka3b	41;861—902	9.7 -7	1.6 -7	17	2
gka3c	61;1891—1952	9.9 -7	8.8 -7	8	14
gka3d	101;5151—5252	1.0 -6	1.2 -5*	30	85*
gka3e	201;20301—20502	1.0 -6	3.6 -5*	161	305*
gka3f	501;125751—126252	1.0 -6	3.7 -5*	2222	3514*
gka4a	81;3321—3402	1.0 -6	5.6 -6*	19	72*
gka4b	51;1326—1377	9.9 -7	5.0 -7	21	2
gka4c	71;2556—2627	1.0 -6	7.6 -6*	14	64*
gka4d	101;5151—5252	1.0 -6	7.2 -6*	19	64*
gka4e	201;20301—20502	1.0 -6	1.2 -5*	187	382*
gka4f	501;125751—126252	1.0 -6	3.4 -5*	2348	3908*
gka5a	51;1326—1377	9.9 -7	6.0 -7	7	10
gka5b	61;1891—1952	9.9 -7	1.4 -7	25	3
gka5c	81;3321—3402	1.0 -6	5.6 -6*	22	36*
gka5d	101;5151—5252	1.0 -6	8.6 -7	19	59
gka5e	201;20301—20502	1.0 -6	4.0 -5*	149	320*
gka5f	501;125751—126252	1.0 -6	1.8 -5*	2210	3798*
gka6a	31;496—527	9.9 -7	9.4 -7	5	6
gka6b	71;2556—2627	9.9 -7	5.3 -8	27	6
gka6c	91;4186—4277	1.0 -6	2.7 -5*	31	69*
gka6d	101;5151—5252	9.9 -7	7.0 -6*	18	46*
gka7a	31;496—527	1.0 -6	9.1 -7	5	4
gka7b	81;3321—3402	9.9 -7	5.2 -7	3	10
gka7c	101;5151—5252	1.0 -6	5.9 -5*	26	62*
gka7d	101;5151—5252	1.0 -6	5.3 -7	16	20
gka8a	101;5151—5252	1.0 -6	4.2 -5*	33	99*
gka8b	91;4186—4277	9.9 -7	8.5 -7	49	17
gka8d	101;5151—5252	1.0 -6	7.0 -7	31	52
gka9b	101;5151—5252	9.5 -7	3.3 -7	46	21
gka9d	101;5151—5252	1.0 -6	7.0 -7	15	38

Table 2.2.7: DSA-BD results on BIQ problems

Instance	n_{s};n_{l}—m	$\langle C, X \rangle$	$b^T y$	Iter	ϵ_P	ϵ_D	Time
be100.1	101;5151—5252	-2.0021326 +4	-2.0021284 +4	2762	4.3 -7	1.0 -6	21
be100.10	101;5151—5252	-1.6408506 +4	-1.6408506 +4	1757	9.8 -7	1.0 -6	13
be100.2	101;5151—5252	-1.7988702 +4	-1.7988693 +4	2249	1.0 -6	6.8 -7	17
be100.3	101;5151—5252	-1.8231054 +4	-1.8231144 +4	2307	4.5 -7	1.0 -6	18
be100.4	101;5151—5252	-1.9841800 +4	-1.9841754 +4	2483	1.0 -6	9.0 -7	19
be100.5	101;5151—5252	-1.6888702 +4	-1.6888698 +4	2158	9.9 -7	1.0 -6	17
be100.6	101;5151—5252	-1.8148225 +4	-1.8148220 +4	1907	9.1 -7	9.9 -7	14
be100.7	101;5151—5252	-1.9700850 +4	-1.9700852 +4	2298	1.0 -6	8.9 -7	17
be100.8	101;5151—5252	-1.9946409 +4	-1.9946451 +4	1759	6.8 -7	9.2 -7	13
be100.9	101;5151—5252	-1.4263373 +4	-1.4263380 +4	2054	1.0 -6	7.4 -7	16
be120.3.1	121;7381—7502	-1.3803561 +4	-1.3803555 +4	2338	9.9 -7	9.7 -7	25
be120.3.10	121;7381—7502	-1.2930857 +4	-1.2930864 +4	1969	9.9 -7	9.0 -7	21
be120.3.2	121;7381—7502	-1.3626630 +4	-1.3626637 +4	2407	1.0 -6	7.6 -7	25
be120.3.3	121;7381—7502	-1.2987898 +4	-1.2987905 +4	1851	1.0 -6	6.8 -7	19
be120.3.4	121;7381—7502	-1.4511234 +4	-1.4511320 +4	1867	7.4 -7	9.8 -7	20
be120.3.5	121;7381—7502	-1.1991909 +4	-1.1991906 +4	2834	9.5 -7	1.0 -6	30
be120.3.6	121;7381—7502	-1.3432058 +4	-1.3432057 +4	2717	1.0 -6	9.3 -7	29
be120.3.7	121;7381—7502	-1.4564113 +4	-1.4564109 +4	4381	8.1 -7	1.0 -6	46
be120.3.8	121;7381—7502	-1.5303024 +4	-1.5303018 +4	3775	1.0 -6	6.7 -7	40
be120.3.9	121;7381—7502	-1.1241319 +4	-1.1241320 +4	3679	1.0 -6	6.6 -7	38
be120.8.1	121;7381—7502	-2.0193959 +4	-2.0193956 +4	1968	9.9 -7	7.5 -7	21
be120.8.10	121;7381—7502	-2.0024004 +4	-2.0024041 +4	2434	1.0 -6	9.8 -7	26
be120.8.2	121;7381—7502	-2.0074132 +4	-2.0074127 +4	3013	1.0 -6	9.4 -7	32
be120.8.3	121;7381—7502	-2.0505900 +4	-2.0505900 +4	2521	1.0 -6	8.5 -7	27
be120.8.4	121;7381—7502	-2.1779801 +4	-2.1779786 +4	2300	1.0 -6	9.2 -7	25
be120.8.5	121;7381—7502	-2.1316282 +4	-2.1316282 +4	2541	1.0 -6	6.1 -7	27
be120.8.6	121;7381—7502	-1.9676959 +4	-1.9676955 +4	2256	9.9 -7	7.4 -7	24
be120.8.7	121;7381—7502	-2.3732388 +4	-2.3732401 +4	1914	9.9 -7	7.4 -7	20
be120.8.8	121;7381—7502	-2.1204746 +4	-2.1204757 +4	1788	8.4 -7	9.7 -7	19
be120.8.9	121;7381—7502	-1.9284427 +4	-1.9284426 +4	1776	7.9 -7	1.0 -6	19
be150.3.1	151;11476—11627	-1.9849179 +4	-1.9849134 +4	2240	7.1 -7	1.0 -6	35
be150.3.10	151;11476—11627	-1.9230920 +4	-1.9230899 +4	3327	1.0 -6	6.8 -7	53
be150.3.2	151;11476—11627	-1.8864852 +4	-1.8864824 +4	2860	7.9 -7	1.0 -6	45
be150.3.3	151;11476—11627	-1.8043715 +4	-1.8043695 +4	2288	1.0 -6	7.9 -7	36
be150.3.4	151;11476—11627	-2.0652667 +4	-2.0652658 +4	2264	1.0 -6	7.1 -7	36
be150.3.5	151;11476—11627	-1.7768649 +4	-1.7768626 +4	2797	1.0 -6	8.3 -7	44
be150.3.7	151;11476—11627	-1.9101308 +4	-1.9101291 +4	2268	7.2 -7	1.0 -6	36
be150.3.8	151;11476—11627	-1.9698060 +4	-1.9698030 +4	3325	6.6 -7	1.0 -6	53
be150.3.9	151;11476—11627	-1.4103367 +4	-1.4103365 +4	2207	9.5 -7	7.5 -7	35
be150.8.1	151;11476—11627	-2.9143686 +4	-2.9143682 +4	2004	1.0 -6	7.2 -7	32
be150.8.10	151;11476—11627	-3.0047974 +4	-3.0047971 +4	2400	1.0 -6	6.8 -7	38
be150.8.2	151;11476—11627	-2.8821096 +4	-2.8821093 +4	2098	1.0 -6	7.2 -7	34
be150.8.3	151;11476—11627	-3.1060369 +4	-3.1060374 +4	2243	1.0 -6	8.5 -7	36
be150.8.4	151;11476—11627	-2.8729295 +4	-2.8729280 +4	2500	1.0 -6	7.4 -7	40
be150.8.5	151;11476—11627	-2.9482071 +4	-2.9482048 +4	2206	7.9 -7	9.9 -7	35
be150.8.6	151;11476—11627	-3.1437234 +4	-3.1437227 +4	2503	1.0 -6	7.4 -7	40
be150.8.7	151;11476—11627	-3.3252109 +4	-3.3252049 +4	2827	8.7 -7	1.0 -6	45
be150.8.8	151;11476—11627	-3.1599994 +4	-3.1599918 +4	3311	9.4 -7	9.8 -7	52
be150.8.9	151;11476—11627	-2.7110727 +4	-2.7110667 +4	2897	1.0 -6	6.9 -7	46
be200.3.1	201;20301—20502	-2.7716091 +4	-2.7716020 +4	2459	1.0 -6	9.2 -7	67
be200.3.10	201;20301—20502	-2.5760689 +4	-2.5760654 +4	3443	1.0 -6	8.4 -7	92
be200.3.2	201;20301—20502	-2.6760792 +4	-2.6760748 +4	2685	9.0 -7	1.0 -6	73
be200.3.3	201;20301—20502	-2.9478639 +4	-2.9478586 +4	5007	1.0 -6	8.8 -7	138
be200.3.4	201;20301—20502	-2.9106208 +4	-2.9106169 +4	3204	1.0 -6	7.6 -7	87
be200.3.5	201;20301—20502	-2.8072990 +4	-2.8072942 +4	4143	9.7 -7	1.0 -6	111
be200.3.6	201;20301—20502	-2.7928345 +4	-2.7928321 +4	2529	1.0 -6	8.3 -7	68
be200.3.7	201;20301—20502	-3.1620504 +4	-3.1620469 +4	3093	8.4 -7	1.0 -6	84
be200.3.8	201;20301—20502	-2.9244286 +4	-2.9244248 +4	2825	1.0 -6	6.7 -7	76
be200.3.9	201;20301—20502	-2.6437048 +4	-2.6436998 +4	5932	1.0 -6	7.0 -7	161
be200.8.1	201;20301—20502	-5.0869496 +4	-5.0869383 +4	4314	8.9 -7	1.0 -6	117
be200.8.10	201;20301—20502	-4.5743067 +4	-4.5742974 +4	2849	1.0 -6	8.3 -7	77
be200.8.2	201;20301—20502	-4.4336053 +4	-4.4335972 +4	2267	9.5 -7	1.0 -6	61
be200.8.3	201;20301—20502	-4.6253974 +4	-4.6253907 +4	3507	1.0 -6	8.1 -7	94
be200.8.4	201;20301—20502	-4.6621241 +4	-4.6621195 +4	2409	1.0 -6	9.2 -7	66
be200.8.5	201;20301—20502	-4.4271235 +4	-4.4271203 +4	2955	1.0 -6	9.5 -7	81
be200.8.6	201;20301—20502	-5.1218884 +4	-5.1218773 +4	3448	1.0 -6	6.7 -7	94
be200.8.7	201;20301—20502	-4.9352768 +4	-4.9352679 +4	2581	1.0 -6	8.2 -7	69
be200.8.8	201;20301—20502	-4.7689168 +4	-4.7689152 +4	2871	1.0 -6	7.1 -7	78
be200.8.9	201;20301—20502	-4.5495602 +4	-4.5495544 +4	3203	1.0 -6	6.8 -7	87
be250.1	251;31626—31877	-2.5119464 +4	-2.5119438 +4	5051	1.0 -6	6.9 -7	209
be250.10	251;31626—31877	-2.4355024 +4	-2.4354984 +4	5625	1.0 -6	8.0 -7	230
be250.2	251;31626—31877	-2.3681493 +4	-2.3681451 +4	4519	1.0 -6	7.3 -7	183
be250.3	251;31626—31877	-2.4000002 +4	-2.3999963 +4	3908	9.3 -7	9.9 -7	161
be250.4	251;31626—31877	-2.5720317 +4	-2.5720252 +4	8912	1.0 -6	7.4 -7	362
be250.5	251;31626—31877	-2.2374710 +4	-2.2374661 +4	4736	7.9 -7	1.0 -6	193
be250.6	251;31626—31877	-2.4018844 +4	-2.4018822 +4	4561	1.0 -6	6.7 -7	188
be250.7	251;31626—31877	-2.5118959 +4	-2.5118946 +4	5022	1.0 -6	7.5 -7	207
be250.8	251;31626—31877	-2.5020398 +4	-2.5020366 +4	4358	1.0 -6	6.8 -7	176
be250.9	251;31626—31877	-2.1397059 +4	-2.1396994 +4	5774	9.3 -7	1.0 -6	239
bqp100-1	101;5151—5252	-8.3803844 +3	-8.3803874 +3	1951	9.9 -7	9.4 -7	15
bqp100-10	101;5151—5252	-1.2980272 +4	-1.2980253 +4	3594	1.0 -6	7.0 -7	27
bqp100-2	101;5151—5252	-1.1489258 +4	-1.1489238 +4	3706	1.0 -6	8.2 -7	28
bqp100-3	101;5151—5252	-1.3153184 +4	-1.3153176 +4	5757	3.9 -7	1.0 -6	43
bqp100-4	101;5151—5252	-1.0731890 +4	-1.0731883 +4	3279	1.0 -6	9.7 -7	25
bqp100-5	101;5151—5252	-9.4870275 +3	-9.4870180 +3	3452	1.0 -6	8.3 -7	26
bqp100-6	101;5151—5252	-1.0824760 +4	-1.0824748 +4	2176	1.0 -6	7.8 -7	16
bqp100-7	101;5151—5252	-1.0689155 +4	-1.0689137 +4	1817	1.0 -6	9.1 -7	14
bqp100-8	101;5151—5252	-1.1769990 +4	-1.1769980 +4	3311	1.0 -6	7.0 -7	25

Table 2.2.8: DSA-BD results on BIQ problems

Instance	$n_s; n_l \vert m$	$\langle C, X \rangle$	$b^T y$	Iter	ϵ_P	ϵ_D	Time
bqp100-9	101;5151—5252	-1.1733253 +4	-1.1733258 +4	4696	2.9 -7	1.0 -6	35
bqp250-1	251;31626—31877	-4.7663112 +4	-4.7662955 +4	4527	8.8 -7	1.0 -6	182
bqp250-10	251;31626—31877	-4.3014529 +4	-4.3014392 +4	3154	1.0 -6	9.9 -7	131
bqp250-2	251;31626—31877	-4.7222380 +4	-4.7222306 +4	4677	1.0 -6	7.2 -7	191
bqp250-3	251;31626—31877	-5.1076751 +4	-5.1076552 +4	5100	5.3 -7	1.0 -6	209
bqp250-4	251;31626—31877	-4.3312555 +4	-4.3312472 +4	3666	8.9 -7	1.0 -6	151
bqp250-5	251;31626—31877	-5.0004327 +4	-5.0004210 +4	6642	9.6 -7	1.0 -6	272
bqp250-6	251;31626—31877	-4.3668862 +4	-4.3668712 +4	5322	8.2 -7	1.0 -6	221
bqp250-7	251;31626—31877	-4.8921743 +4	-4.8921564 +4	4765	1.0 -6	8.6 -7	193
bqp250-8	251;31626—31877	-3.8779549 +4	-3.8779496 +4	3473	6.8 -7	1.0 -6	144
bqp250-9	251;31626—31877	-5.1497554 +4	-5.1497497 +4	4650	1.0 -6	9.0 -7	195
bqp50-1	51;1326—1377	-2.1439177 +3	-2.1439146 +3	2795	1.0 -6	6.9 -7	9
bqp50-10	51;1326—1377	-3.6263711 +3	-3.6263710 +3	1938	9.1 -7	9.5 -7	6
bqp50-2	51;1326—1377	-3.7425229 +3	-3.7425060 +3	6076	9.9 -7	9.2 -7	20
bqp50-3	51;1326—1377	-4.6372395 +3	-4.6372264 +3	2220	4.7 -7	1.0 -6	7
bqp50-4	51;1326—1377	-3.5839746 +3	-3.5839694 +3	12719	9.7 -7	1.0 -6	42
bqp50-5	51;1326—1377	-4.0776076 +3	-4.0776061 +3	2716	1.0 -6	8.7 -7	9
bqp50-6	51;1326—1377	-3.7125866 +3	-3.6959056 +3	20000	6.4 -4	1.4 -3	69
bqp50-7	51;1326—1377	-4.6496912 +3	-4.6496901 +3	2218	4.3 -7	1.0 -6	7
bqp50-8	51;1326—1377	-4.2692350 +3	-4.2692377 +3	2073	9.6 -7	9.6 -7	7
bqp50-9	51;1326—1377	-3.9216432 +3	-3.9216406 +3	1792	1.0 -6	9.9 -7	6
bqp500-1	501;125751—126252	-1.2596423 +5	-1.2596374 +5	7353	7.0 -7	1.0 -6	1230
bqp500-10	501;125751—126252	-1.3853448 +5	-1.3853384 +5	7411	7.9 -7	1.0 -6	1241
bqp500-2	501;125751—126252	-1.3601108 +5	-1.3601080 +5	8252	1.0 -6	7.5 -7	1364
bqp500-3	501;125751—126252	-1.3845346 +5	-1.3845287 +5	7329	7.7 -7	1.0 -6	1232
bqp500-4	501;125751—126252	-1.3932842 +5	-1.3932790 +5	7910	6.8 -7	1.0 -6	1303
bqp500-5	501;125751—126252	-1.3409217 +5	-1.3409177 +5	7790	1.0 -6	8.2 -7	1294
bqp500-6	501;125751—126252	-1.3076439 +5	-1.3076411 +5	7363	7.3 -7	1.0 -6	1236
bqp500-7	501;125751—126252	-1.3149149 +5	-1.3149108 +5	7621	6.9 -7	1.0 -6	1280
bqp500-8	501;125751—126252	-1.3348988 +5	-1.3348960 +5	7911	1.0 -6	9.8 -7	1314
bqp500-9	501;125751—126252	-1.3028828 +5	-1.3028790 +5	7166	1.0 -6	9.5 -7	1187
gka10b	126;8001—8127	-1.5557650 +2	-1.5555950 +2	1822	6.8 -7	1.0 -6	22
gka10d	101;5151—5252	-2.0108576 +4	-2.0108575 +4	2008	1.0 -6	8.9 -7	17
gka1a	51;1326—1377	-3.5374674 +3	-3.5366291 +3	20000	9.4 -6	3.4 -5	74
gka1b	21;231—252	-1.3300000 +2	-1.3300000 +2	1130	8.9 -7	9.2 -7	3
gka1c	41;861—902	-5.1138290 +3	-5.1138227 +3	3866	5.4 -7	1.0 -6	11
gka1d	101;5151—5252	-6.5284315 +3	-6.5283639 +3	3937	1.0 -6	9.9 -7	33
gka1e	201;20301—20502	-1.7069817 +4	-1.7069805 +4	4971	1.0 -6	7.2 -7	181
gka1f	501;125751—126252	-6.5559070 +4	-6.5558956 +4	7631	1.0 -6	7.7 -7	2195
gka2b	31;496—527	-1.2130600 +2	-1.2129990 +2	2220	1.0 -6	7.3 -7	6
gka2c	51;1326—1377	-6.3200104 +3	-6.3199986 +3	2136	5.2 -7	9.9 -7	7
gka2d	101;5151—5252	-6.9907097 +3	-6.9907100 +3	2042	7.1 -7	1.0 -6	17
gka2e	201;20301—20502	-2.4917636 +4	-2.4917596 +4	3231	1.0 -6	7.4 -7	121
gka2f	501;125751—126252	-1.0793177 +5	-1.0793138 +5	8508	1.0 -6	9.4 -7	2488
gka3a	71;2556—2627	-6.3859990 +3	-6.3859859 +3	2202	6.4 -7	9.9 -7	11
gka3b	41;861—902	-1.1799980 +2	-1.1801510 +2	6358	9.7 -7	9.5 -7	17
gka3c	61;1891—1952	-6.8138990 +3	-6.8138886 +3	1988	3.5 -7	9.9 -7	8
gka3d	101;5151—5252	-9.7343322 +3	-9.7343347 +3	3600	1.0 -6	8.7 -7	30
gka3e	201;20301—20502	-2.6898741 +4	-2.6898705 +4	4302	1.0 -6	6.8 -7	161
gka3f	501;125751—126252	-1.5015105 +5	-1.5015061 +5	7522	9.5 -7	1.0 -6	2222
gka4a	81;3321—3402	-8.8809678 +3	-8.8809583 +3	2936	2.8 -7	1.0 -6	19
gka4b	51;1326—1377	-1.2899980 +2	-1.2902000 +2	6382	9.9 -7	9.8 -7	21
gka4c	71;2556—2627	-7.5650115 +3	-7.5650110 +3	2735	8.6 -7	1.0 -6	14
gka4d	101;5151—5252	-1.1278414 +4	-1.1278415 +4	2279	1.0 -6	7.6 -7	19
gka4e	201;20301—20502	-3.7225147 +4	-3.7225072 +4	4524	9.4 -7	1.0 -6	187
gka4f	501;125751—126252	-1.8708790 +5	-1.8708748 +5	7948	1.0 -6	7.9 -7	2348
gka5a	51;1326—1377	-5.8970505 +3	-5.8970492 +3	2050	3.2 -7	9.9 -7	7
gka5b	61;1891—1952	-1.4999980 +2	-1.5002340 +2	6393	9.8 -7	9.9 -7	25
gka5c	81;3321—3402	-7.5762319 +3	-7.5762289 +3	3702	1.0 -6	9.7 -7	22
gka5d	101;5151—5252	-1.2398864 +4	-1.2398866 +4	2234	1.0 -6	6.8 -7	19
gka5e	201;20301—20502	-3.8002313 +4	-3.8002274 +4	3945	1.0 -6	8.5 -7	149
gka5f	501;125751—126252	-2.0691429 +5	-2.0691388 +5	7445	7.0 -7	1.0 -6	2210
gka6a	31;496—527	-4.1032065 +3	-4.1032066 +3	1951	1.8 -7	9.9 -7	5
gka6b	71;2556—2627	-1.4600020 +2	-1.4597220 +2	5673	9.9 -7	9.7 -7	27
gka6c	91;4186—4277	-5.9619429 +3	-5.9619530 +3	4360	1.0 -6	8.9 -7	31
gka6d	101;5151—5252	-1.4929358 +4	-1.4929358 +4	2131	7.3 -7	9.9 -7	18
gka7a	31;496—527	-4.6386078 +3	-4.6386032 +3	2134	1.0 -6	6.4 -7	5
gka7b	81;3321—3402	-1.6035690 +2	-1.6035880 +2	493	9.2 -7	9.9 -7	3
gka7c	101;5151—5252	-7.3164493 +3	-7.3164551 +3	3144	9.0 -7	1.0 -6	26
gka7d	101;5151—5252	-1.5375790 +4	-1.5375801 +4	1850	9.5 -7	1.0 -6	16
gka8a	101;5151—5252	-1.1197217 +4	-1.1197215 +4	3678	7.2 -7	1.0 -6	33
gka8b	91;4186—4277	-1.4499980 +2	-1.4503580 +2	6797	9.9 -7	9.7 -7	49
gka8d	101;5151—5252	-1.7005361 +4	-1.7005352 +4	3643	6.9 -7	1.0 -6	31
gka9b	101;5151—5252	-1.3700010 +2	-1.3696100 +2	6155	9.5 -7	9.4 -7	46
gka9d	101;5151—5252	-1.6533903 +4	-1.6533898 +4	1776	9.6 -7	1.0 -6	15

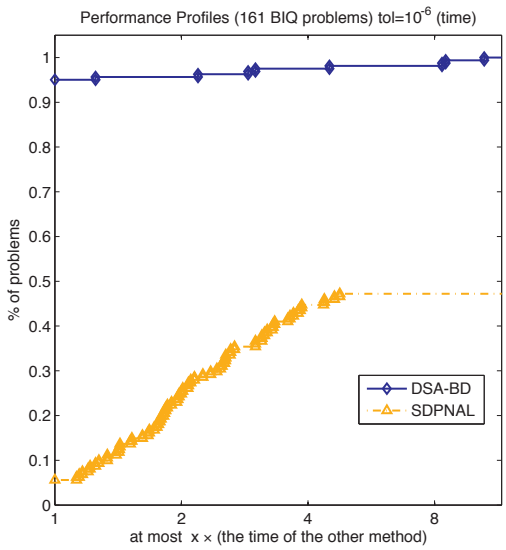

Figure 2.2.4: Performance profiles of DSA-BD and SDPNAL for solving 161 SDP relaxations of BIQ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

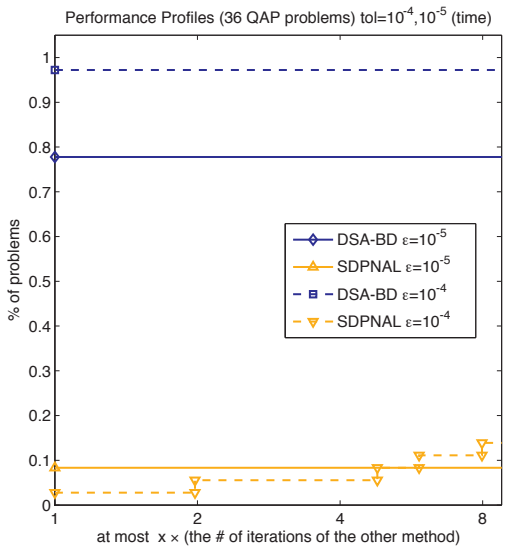

Figure 2.2.5: Performance profiles of DSA-BD and SDPNAL for solving 36 SDP relaxations of QAPs with accuracies $\bar{\epsilon} = 10^{-4}, 10^{-5}$.

xx^T. In [53], it is shown that an SDP relaxation of the QAP is

$$\begin{aligned} \max \quad & \langle B \otimes A, Y \rangle \\ \text{s.t.} \quad & \sum_{i=1}^{n} Y^{ii} = I, \ \langle I, Y^{ij} \rangle = \delta_{ij} \ \forall \ 1 \le i \le j \le n, && \text{(2.2.27a)} \\ & \langle E, Y^{ij} \rangle = 1 \ , \forall 1 \le i \le j \le n, && \text{(2.2.27b)} \\ & Y \succeq 0, \ Y \ge 0, && \text{(2.2.27c)} \end{aligned}$$

where $E \in \mathbb{R}^{n \times n}$ is the matrix of ones, and $\delta_{ij} = 1$ if $i = j$, and 0 otherwise.

Table 2.2.9 compares the two methods on a collection of SDP relaxations of QAPs using the tolerances $\bar{\epsilon} = 10^{-4}, 10^{-5}$. In this table, computational results for each instance are reported in two rows, the first one for $\bar{\epsilon} = 10^{-4}$, and the second one for $\bar{\epsilon} = 10^{-5}$. Table 2.2.10 gives more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals.

Figure 2.2.5 plots the performance profiles of both methods based on these SDP relaxations of QAPs. Note that SDPNAL fails to find a solution with an accuracy of at least

10^{-4} for almost all of the SDP relaxations of QAPs tested. On the other hand, our method DSA-BD was able to find a solution with an accuracy of at least 10^{-5} for almost all of the SDP relaxations of QAPs tested. Observe also that a flat line in this figure means that the corresponding method is faster for some instances, but fails to solve the rest of them. For example, SDPNAL is the fastest method for solving $\sim 8\%$ of the instances with an accuracy of $\bar{\epsilon} = 10^{-5}$, but fails to obtain a solution for the other $\sim 92\%$ of the instances.

2.2.2.5 SDPs arising from relaxation of maximum stable set problems

This subsection compares the performance of our method DSA-BD with that of BP and SDPNAL on a collection of SDPs corresponding to θ-functions and θ_+-functions of graph stable set problems.

The SDPs for θ-functions and θ_+-functions of graph stable set problems can be described as follows. Given a graph G with n nodes and an edge set E, the SDP relaxations $\theta(G)$ and $\theta_+(G)$ of the maximum stable set problem are defined as

$$\begin{aligned}
\theta(G) := \max\, & C \bullet X & \theta_+(G) := \max\, & C \bullet X \\
\text{s.t } & X \succeq 0, & \text{s.t } & X \succeq 0, && \text{(2.2.28a)} \\
& I \bullet X = 1, & & I \bullet X = 1, && \text{(2.2.28b)} \\
& X_{ij} = 0,\ (i,j) \in E, & & X_{ij} = 0,\ (i,j) \in E,\ X \geq 0, && \text{(2.2.28c)}
\end{aligned}$$

where $C = ee^T$, $X \in \mathcal{S}^n$ and $e \in \mathbb{R}^n$ is the vector with all ones.

Tables 2.2.11 and 2.2.13 compare the three methods on a collection of $\theta(G)$ and $\theta_+(G)$ problems using the tolerance $\bar{\epsilon} = 10^{-6}$. Tables 2.2.12 and 2.2.14 give more detailed computational results on these instances obtained by our method DSA-BD, such as the objective values, number of iterations, and the primal and dual relative residuals. Figures 2.2.6 and 2.2.7 plot the performance profiles of the three methods based on these $\theta(G)$ and $\theta_+(G)$ problems.

Note that even though SDPNAL is faster than BP and DSA-BD on more than 60% of the $\theta(G)$ instances, BP and DSA-BD are more robust, as they are able to solve almost all

Table 2.2.9: Comparison of the methods on QAPs

Instance	Problem $n_s; n_l \vert m$	$\max\{\epsilon_P, \epsilon_D\}$ DSA-BD	$\max\{\epsilon_P, \epsilon_D\}$ SDPNAL	Time DSA-BD	Time SDPNAL
bur26a	676;228826—229877	1.0 -4	1.4 -4*	453	4669*
		1.0 -5	1.4 -4*	13845	4669*
bur26b	676;228826—229877	1.0 -4	1.5 -4*	600	3577*
		1.0 -5	1.5 -4*	13330	3577*
bur26c	676;228826—229877	1.0 -4	2.4 -4*	641	5790*
		1.0 -5	2.4 -4*	14684	5790*
bur26d	676;228826—229877	1.0 -4	2.5 -4*	735	4436*
		1.0 -5	2.5 -4*	16765	4436*
bur26e	676;228826—229877	1.0 -4	2.3 -4*	750	6240*
		1.0 -5	2.3 -4*	5168	6240*
bur26f	676;228826—229877	1.0 -4	2.1 -4*	856	4509*
		1.0 -5	2.1 -4*	10088	4509*
bur26g	676;228826—229877	1.0 -4	1.9 -4*	311	4115*
		1.0 -5	1.9 -4*	6814	4115*
bur26h	676;228826—229877	9.9 -5	3.4 -5	195	1552
		1.0 -5	2.5 -5*	1284	4419*
chr12a	144;10440—10672	1.1 -4*	8.0 -5	2664*	106
		1.1 -4*	4.8 -6	2664*	110
chr12b	144;10440—10672	8.9 -5	1.6 -5	27	132
		4.3 -5*	9.2 -6	2762*	144
chr12c	144;10440—10672	9.9 -5	1.5 -4*	42	284*
		9.7 -6	1.5 -4*	769	284*
chr15a	225;25425—25783	1.0 -4	2.8 -4*	252	656*
		1.0 -5	2.8 -4*	1947	656*
chr15b	225;25425—25783	1.0 -4	4.4 -4*	129	325*
		2.4 -5*	4.4 -4*	5768*	325*
chr15c	225;25425—25783	1.0 -4	4.7 -7	168	331
		6.5 -5*	4.7 -7	5534*	331
chr18a	324;52650—53161	9.9 -5	7.0 -4*	477	1475*
		1.0 -5	7.0 -4*	3488	1475*
chr18b	324;52650—53161	1.0 -4	4.0 -5	54	314
		1.0 -5	4.0 -5*	101	797*
chr20a	400;80200—80828	1.0 -4	4.1 -4*	522	2133*
		1.0 -5	4.1 -4*	1109	2133*
chr20b	400;80200—80828	1.0 -4	1.4 -4*	941	1736*
		2.6 -5*	1.4 -4*	17235*	1736*
chr20c	400;80200—80828	1.0 -4	4.3 -4*	650	1622*
		9.8 -6	4.3 -4*	5365	1622*
chr22a	484;117370—118127	1.0 -4	3.1 -4*	302	2911*
		1.2 -5*	3.1 -4*	25802*	2911*
chr22b	484;117370—118127	1.0 -4	2.2 -4*	308	2273*
		1.3 -5*	2.2 -4*	24400*	2273*
nug12	144;10440—10672	1.0 -4	1.6 -4*	31	46*
		1.0 -5	1.6 -4*	239	46*
nug14	196;19306—19619	1.0 -4	3.1 -4*	91	119*
		1.0 -5	3.1 -4*	1144	119*
nug15	225;25425—25783	1.0 -4	2.2 -4*	108	159*
		1.0 -5	2.2 -4*	1019	159*
nug16a	256;32896—33302	1.0 -4	1.8 -4*	203	336*
		1.0 -5	1.8 -4*	2514	336*
nug16b	256;32896—33302	1.0 -4	2.7 -4*	103	194*
		1.0 -5	2.7 -4*	914	194*
nug17	289;41905—42362	1.0 -4	1.6 -4*	185	272*
		1.0 -5	1.6 -4*	1742	272*
nug18	324;52650—53161	1.0 -4	2.2 -4*	206	297*
		1.0 -5	2.2 -4*	1827	297*
nug20	400;80200—80828	1.0 -4	1.8 -4*	299	471*
		1.0 -5	1.8 -4*	2466	471*
nug21	441;97461—98152	1.0 -4	2.2 -4*	420	665*
		1.0 -5	2.2 -4*	3780	665*
nug21	441;97461—98152	1.0 -4	2.2 -4*	418	663*
		1.0 -5	2.2 -4*	3814	663*
nug22	484;117370—118127	1.0 -4	2.3 -4*	659	961*
		1.0 -5	2.3 -4*	5562	961*
nug22	484;117370—118127	1.0 -4	2.3 -4*	651	1030*
		1.0 -5	2.3 -4*	5473	1030*
tai25a	625;195625—196598	1.0 -4	1.4 -4*	471	1035*
		1.0 -5	1.4 -4*	3688	1035*
tai25b	625;195625—196598	1.0 -4	5.1 -4*	1891	2438*
		1.2 -5*	5.1 -4*	28402*	2438*
tai30a	900;405450—406843	1.0 -4	1.0 -4*	1059	2452*
		1.0 -5	1.0 -4*	8223	2452*

Table 2.2.10: DSA-BD results on QAPs

INSTANCE	$n_s; n_l \vert m$	$\langle C, X \rangle$	$b^T y$	ITER	ϵ_P	ϵ_D	TIME
bur26a	676;228826—229877	5.4263208 +6	5.4268401 +6	50000	5.6 -6	7.0 -6	22578
bur26b	676;228826—229877	3.8172495 +6	3.8176224 +6	50000	6.2 -6	6.1 -6	23085
bur26c	676;228826—229877	5.4270012 +6	5.4277912 +6	50000	7.9 -6	9.4 -6	22290
bur26d	676;228826—229877	3.8206639 +6	3.8211360 +6	50000	6.4 -6	8.2 -6	22612
bur26e	676;228826—229877	5.3874085 +6	5.3877112 +6	50000	3.2 -6	4.3 -6	22577
bur26f	676;228826—229877	3.7820836 +6	3.7821498 +6	47127	6.6 -7	1.0 -6	21173
bur26g	676;228826—229877	1.0117250 +7	1.0117274 +7	32438	1.0 -6	3.9 -7	14530
bur26h	676;228826—229877	7.0986748 +6	7.0987800 +6	18086	9.9 -7	7.3 -7	7645
chr12a	144;10440—10672	9.5602473 +3	9.6751178 +3	100000	2.7 -5	1.1 -4	2664
chr12b	144;10440—10672	9.7420318 +3	9.8474864 +3	100000	1.3 -5	4.3 -5	2762
chr12c	144;10440—10672	1.1156025 +4	1.1163345 +4	100000	5.4 -7	3.2 -6	2578
chr15a	225;25425—25783	9.8957979 +3	9.9101536 +3	100000	5.5 -6	4.4 -6	5152
chr15b	225;25425—25783	7.9898869 +3	7.8800531 +3	100000	3.3 -6	2.4 -5	5768
chr15c	225;25425—25783	9.5039347 +3	9.2142006 +3	100000	1.3 -5	6.5 -5	5534
chr18a	324;52650—53161	1.1097999 +4	1.1085506 +4	100000	2.2 -7	1.8 -6	10592
chr18b	324;52650—53161	1.5339652 +3	1.5340022 +3	1493	9.1 -7	1.0 -6	149
chr20a	400;80200—80828	2.1919994 +3	2.1902299 +3	100000	4.0 -7	1.8 -6	16496
chr20b	400;80200—80828	2.2980107 +3	2.2705528 +3	100000	3.9 -6	2.6 -5	17235
chr20c	400;80200—80828	1.4144561 +4	1.4156713 +4	100000	6.2 -7	1.6 -6	16440
chr22a	484;117370—118127	6.1565369 +3	6.1744853 +3	100000	6.1 -6	1.2 -5	25802
chr22b	484;117370—118127	6.1946119 +3	6.2137604 +3	100000	6.1 -6	1.3 -5	24400
nug12	144;10440—10672	5.6778580 +2	5.6788770 +2	100000	1.5 -6	1.2 -6	1776
nug14	196;19306—19619	1.0095760 +3	1.0098620 +3	100000	4.2 -6	2.8 -6	2877
nug15	225;25425—25783	1.1399857 +3	1.1402761 +3	100000	2.8 -6	2.1 -6	3692
nug16a	256;32896—33302	1.5983130 +3	1.5988500 +3	100000	5.5 -6	3.9 -6	4671
nug16b	256;32896—33302	1.2176912 +3	1.2179803 +3	100000	2.1 -6	1.6 -6	4388
nug17	289;41905—42362	1.7062557 +3	1.7066925 +3	100000	3.0 -6	2.3 -6	5733
nug18	324;52650—53161	1.8927180 +3	1.8931364 +3	100000	2.6 -6	1.8 -6	7055
nug20	400;80200—80828	2.5054307 +3	2.5058998 +3	100000	2.1 -6	1.7 -6	11029
nug21	441;97461—98152	2.3808229 +3	2.3813933 +3	100000	2.7 -6	2.0 -6	13311
nug21	441;97461—98152	2.3808229 +3	2.3813933 +3	100000	2.7 -6	2.0 -6	13435
nug22	484;117370—118127	3.5266774 +3	3.5277129 +3	100000	3.4 -6	2.4 -6	16183
nug22	484;117370—118127	3.5266774 +3	3.5277129 +3	100000	3.4 -6	2.4 -6	15980
tai25a	625;195625—196598	1.0964898 +6	1.0965735 +6	100000	1.3 -6	8.4 -7	28954
tai25b	625;195625—196598	3.3674678 +8	3.3752145 +8	100000	1.0 -5	1.2 -5	28402
tai30a	900;405450—406843	1.7066136 +6	1.7067425 +6	100000	1.2 -6	8.9 -7	63839

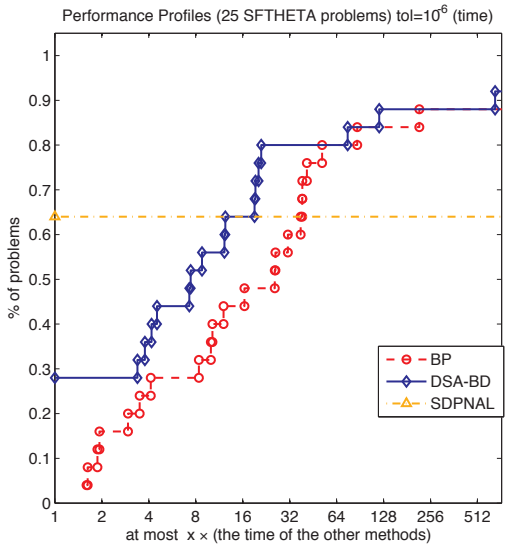

Figure 2.2.6: Performance profiles of DSA-BD, SDPNAL and BP for solving 25 $\theta(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

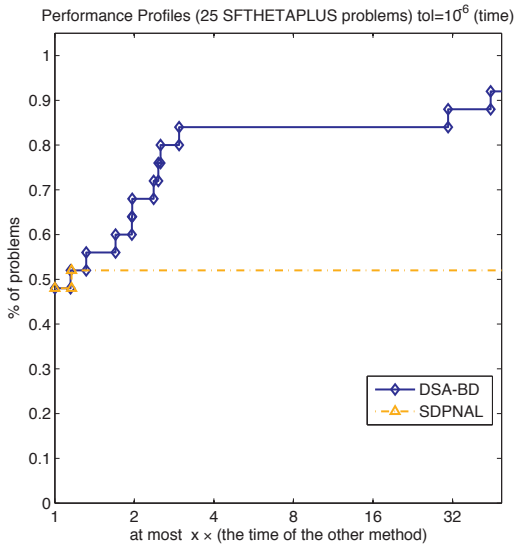

Figure 2.2.7: Performance profiles of DSA-BD and SDPNAL for solving 25 $\theta_+(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

Table 2.2.11: Comparison of the methods on $\theta(G)$

Problem		$\max\{\epsilon_P, \epsilon_D\}$			Time		
Instance	$n_s \vert m$	BP	DSA-BD	SDPNAL	BP	DSA-BD	SDPNAL
1dc.1024	1024—24064	1.00 -6	1.00 -6	1.62 -6*	10029	5192	585*
1dc.512	512—9728	1.00 -6	1.00 -6	2.94 -6*	2258	1205	152*
1et.1024	1024—9601	1.00 -6	1.00 -6	2.51 -6*	17026	4129	862*
1et.512	512—4033	1.00 -6	9.69 -7	7.03 -7	1880	529	60
1tc.1024	1024—7937	1.09 -6*	9.95 -7	2.70 -6*	22786*	6996	1322*
1tc.512	512—3265	9.99 -7	1.00 -6	8.94 -6*	4645	1577	337*
1zc.1024	1024—16641	9.27 -7	9.66 -7	8.71 -7	911	434	35
1zc.512	512—6913	9.14 -7	9.53 -7	8.56 -7	138	62	8
2dc.1024	1024—169163	9.97 -7	9.92 -7	6.47 -6*	15086	9297	1638*
2dc.512	512—54896	9.97 -7	9.68 -7	1.51 -5*	1856	1162	448*
G43	1000—9991	8.68 -7	9.67 -7	7.51 -7	1347	665	35
G44	1000—9991	9.32 -7	9.64 -7	6.61 -7	1340	703	35
G45	1000—9991	9.98 -7	9.76 -7	5.70 -7	1356	696	36
G46	1000—9991	9.60 -7	9.67 -7	6.78 -7	1384	706	34
G47	1000—9991	8.50 -7	9.33 -7	6.36 -7	1365	651	53
G51	1000—5910	9.99 -7	9.99 -7	9.74 -7	2976	3217	852
G52	1000—5917	2.76 -6*	3.07 -6*	1.09 -5*	11794*	9157*	1591*
G53	1000—5915	1.64 -5*	3.40 -6*	9.78 -6*	12053*	8995*	1199*
G54	1000—5917	9.99 -7	9.99 -7	5.94 -7	4875	2149	476
hamming-10-2	1024—23041	8.92 -7	9.62 -7	1.56 -7	837	1219	16
hamming-9-5-6	512—53761	9.55 -7	9.74 -7	2.15 -8	250	346	3
hamming-9-8	512—2305	8.73 -7	9.85 -7	2.02 -8	583	1785	3
theta12	600—17979	9.63 -7	8.84 -7	7.28 -7	163	98	13
theta123	600—90020	9.96 -7	9.81 -7	2.68 -7	160	65	19
theta162	800—127600	9.93 -7	9.84 -7	4.03 -7	306	128	31

Table 2.2.12: DSA-BD results on $\theta(G)$

Instance	$n_s \vert m$	$\langle C, X \rangle$	$b^T y$	Iter	ϵ_P	ϵ_D	Time
1dc.1024	1024—24064	-9.598590 +1	-9.598590 +1	11431	1.00 -6	5.48 -7	5192
1dc.512	512—9728	-5.303110 +1	-5.303110 +1	11084	1.00 -6	2.91 -7	1205
1et.1024	1024—9601	-1.842274 +2	-1.842274 +2	9751	1.00 -6	2.99 -7	4129
1et.512	512—4033	-1.044244 +2	-1.044244 +2	4770	5.03 -7	9.69 -7	529
1tc.1024	1024—7937	-2.063055 +2	-2.063055 +2	15907	9.70 -7	9.95 -7	6996
1tc.512	512—3265	-1.134009 +2	-1.134009 +2	15003	1.00 -6	3.38 -7	1577
1zc.1024	1024—16641	-1.286670 +2	-1.286670 +2	1059	9.66 -7	8.75 -7	434
1zc.512	512—6913	-6.875000 +1	-6.875000 +1	615	9.53 -7	4.94 -7	62
2dc.1024	1024—169163	-1.863900 +1	-1.863900 +1	18519	9.92 -7	9.79 -7	9297
2dc.512	512—54896	-1.176820 +1	-1.176820 +1	9762	9.61 -7	9.68 -7	1162
G43	1000—9991	-2.806257 +2	-2.806257 +2	1477	9.21 -7	9.67 -7	665
G44	1000—9991	-2.805837 +2	-2.805837 +2	1561	9.20 -7	9.64 -7	703
G45	1000—9991	-2.801862 +2	-2.801862 +2	1556	9.76 -7	9.74 -7	696
G46	1000—9991	-2.798379 +2	-2.798379 +2	1576	9.67 -7	8.40 -7	706
G47	1000—9991	-2.818943 +2	-2.818943 +2	1424	9.33 -7	8.33 -7	651
G51	1000—5910	-3.490001 +2	-3.490001 +2	7163	9.99 -7	2.74 -7	3217
G52	1000—5917	-3.484016 +2	-3.483917 +2	20000	3.07 -6	1.46 -6	9157
G53	1000—5915	-3.483613 +2	-3.483514 +2	20000	3.40 -6	3.39 -6	8995
G54	1000—5917	-3.410002 +2	-3.410002 +2	4732	9.99 -7	6.72 -7	2149
hamming-10-2	1024—23041	-1.024005 +2	-1.024005 +2	2913	9.32 -7	9.62 -7	1219
hamming-9-5-6	512—53761	-8.533340 +1	-8.533340 +1	3219	4.60 -7	9.74 -7	346
hamming-9-8	512—2305	-2.239997 +2	-2.239997 +2	18818	6.00 -7	9.85 -7	1785
theta12	600—17979	-9.280180 +1	-9.280180 +1	613	8.57 -7	8.84 -7	98
theta123	600—90020	-2.466880 +1	-2.466880 +1	358	9.81 -7	8.87 -7	65
theta162	800—127600	-3.700990 +1	-3.700990 +1	398	9.84 -7	8.58 -7	128

of the $\theta(G)$ instances to an accuracy of at least 10^{-6}, while SDPNAL fails to do so in more than 35% of them. Also, BP takes more time than DSA-BD to find a solution with an accuracy of at least 10^{-6} in almost all of the $\theta(G)$ instances tested.

Note also that our method DSA-BD was able to find a solution with an accuracy of at least 10^{-6} for almost all of the $\theta_+(G)$ instances tested, while SDPNAL fails to do so for almost half of them.

Table 2.2.13: Comparison of the methods on $\theta_+(G)$

Problem		$\max\{\epsilon_P, \epsilon_D\}$		Time	
Instance	$n_s \vert m$	DSA-BD	SDPNAL	DSA-BD	SDPNAL
1dc.1024	1024—548864	1.00 -6	3.84 -6*	2882	3563*
1dc.512	512—141056	9.99 -7	2.75 -6*	532	619*
1et.1024	1024—534401	9.98 -7	5.31 -6*	2066	5346*
1et.512	512—135361	9.98 -7	2.08 -5*	347	1145*
1tc.1024	1024—532737	1.00 -6	1.09 -4*	6816	7962*
1tc.512	512—134593	9.99 -7	6.06 -5*	644	1791*
1zc.1024	1024—541441	9.78 -7	2.70 -7	1256	740
1zc.512	512—138241	9.33 -7	9.22 -7	241	82
2dc.1024	1024—693963	1.00 -6	1.40 -4*	1741	8027*
2dc.512	512—186224	1.00 -6	1.19 -4*	564	1954*
G43	1000—510491	8.35 -7	6.92 -7	1310	668
G44	1000—510491	9.92 -7	6.35 -7	1404	594
G45	1000—510491	9.45 -7	8.36 -7	1414	563
G46	1000—510491	9.85 -7	9.13 -7	1450	588
G47	1000—510491	9.38 -7	6.86 -7	1178	602
G51	1000—506410	1.00 -6	5.89 -4*	6796	10261*
G52	1000—506417	1.00 -6	2.84 -4*	7826	10501*
G53	1000—506415	1.19 -6*	2.19 -4*	14918*	9885*
G54	1000—506417	9.99 -7	3.69 -4*	3430	7488*
hamming-10-2	1024—547841	9.87 -7	3.78 -7	2575	58
hamming-9-5-6	512—185089	9.62 -7	4.94 -7	538	17
hamming-9-8	512—133633	1.94 -2*	2.36 -7	3224*	5
theta12	600—198279	9.94 -7	6.42 -7	145	110
theta123	600—270320	9.99 -7	9.02 -7	110	96
theta162	800—448000	9.76 -7	9.34 -7	222	257

Table 2.2.14: DSA-BD results on $\theta_+(G)$

Instance	$n_s \vert m$	$\langle C, X \rangle$	$b^T y$	Iter	ϵ_P	ϵ_D	Time
1dc.1024	1024—548864	-9.555200 +1	-9.555200 +1	3982	1.00 -6	3.74 -7	2882
1dc.512	512—141056	-5.269540 +1	-5.269540 +1	3058	9.99 -7	2.99 -7	532
1et.1024	1024—534401	-1.820735 +2	-1.820735 +2	2964	9.98 -7	8.04 -7	2066
1et.512	512—135361	-1.035492 +2	-1.035492 +2	2144	9.91 -7	9.98 -7	347
1tc.1024	1024—532737	-2.042061 +2	-2.042061 +2	9192	1.00 -6	4.35 -7	6816
1tc.512	512—134593	-1.125343 +2	-1.125343 +2	3807	9.99 -7	8.42 -7	644
1zc.1024	1024—541441	-1.280007 +2	-1.280007 +2	1832	9.31 -7	9.78 -7	1256
1zc.512	512—138241	-6.800010 +1	-6.800010 +1	1462	9.33 -7	5.96 -7	241
2dc.1024	1024—693963	-1.771020 +1	-1.771020 +1	2332	8.99 -7	1.00 -6	1741
2dc.512	512—186224	-1.138370 +1	-1.138370 +1	3107	1.00 -6	3.57 -7	564
G43	1000—510491	-2.797359 +2	-2.797359 +2	1824	7.91 -7	8.35 -7	1310
G44	1000—510491	-2.797468 +2	-2.797468 +2	1963	7.82 -7	9.92 -7	1404
G45	1000—510491	-2.793186 +2	-2.793186 +2	1977	9.45 -7	8.39 -7	1414
G46	1000—510491	-2.790333 +2	-2.790333 +2	1979	7.78 -7	9.85 -7	1450
G47	1000—510491	-2.808927 +2	-2.808927 +2	1657	9.38 -7	8.44 -7	1178
G51	1000—506410	-3.490002 +2	-3.490002 +2	9269	1.00 -6	3.96 -7	6796
G52	1000—506417	-3.483868 +2	-3.483868 +2	10655	1.00 -6	6.93 -7	7826
G53	1000—506415	-3.482135 +2	-3.482114 +2	20000	8.85 -7	1.19 -6	14918
G54	1000—506417	-3.410004 +2	-3.410004 +2	4776	9.44 -7	9.99 -7	3430
hamming-10-2	1024—547841	-8.533390 +1	-8.533390 +1	3626	8.87 -7	9.87 -7	2575
hamming-9-5-6	512—185089	-5.866660 +1	-5.866660 +1	2973	3.85 -7	9.62 -7	538
hamming-9-8	512—133633	-1.644936 +2	-2.693922 +2	20000	1.94 -2	1.12 -2	3224
theta12	600—198279	-9.209090 +1	-9.209090 +1	583	9.94 -7	9.61 -7	145
theta123	600—270320	-2.449530 +1	-2.449530 +1	399	9.99 -7	9.80 -7	110
theta162	800—448000	-3.671160 +1	-3.671160 +1	445	9.76 -7	9.44 -7	222

2.3 A two-easy-block method for conic programming

In this section, we introduce a first-order BD method for minimizing the sum of a convex differentiable function with Lipschitz continuous gradient, and two other proper closed convex (possibly, nonsmooth) functions with easily computable resolvents. Subsection 2.3.1 presents a first-order instance of the A-BD-HPE framework, and corresponding iteration-complexity results, for composite convex optimization problems. Subsection 2.3.2 discusses the specialization of the method of Subsection 2.3.1 to the context of conic optimization problems with a two-easy-block structure. Subsection 2.3.3 describes a practical variant of the BD method of Subsection 2.3.2 which incorporates a dynamic update of the scaling factor to balance the blocks. Section 2.3.4 presents numerical results comparing the latter variant of the BD method to the method discussed in [67]. Section 2.3.5 briefly compares this variant of the BD method with DSA-BD and the method in [68].

2.3.1 A BD algorithm for a class of structured convex optimization

This subsection presents a first-order BD algorithm, and corresponding complexity results, for solving a minimization problem whose objective function is the sum of a finite everywhere convex function with Lipschitz continuous gradient and two proper closed convex (possibly, nonsmooth) functions with easily computable resolvents.

We are concerned with the optimization problem

$$\begin{aligned} \min \quad & f(x) + h_1(x) + h_2(x) \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned} \tag{2.3.1}$$

where:

B.1) $f, h_1, h_2 : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ are convex lower semicontinuous proper functions;

B.2) f is differentiable on $\mathcal{X}$ and its gradient is L_f-Lipschitz continuous, that is,

$$\|\nabla f(x) - \nabla f(x')\| \leq L_f \left\|x - x'\right\|, \qquad \forall x, x' \in \mathcal{X}; \tag{2.3.2}$$

B.3) the intersection of the relative interiors of the effective domains of h_1 and h_2 is nonempty.

In view of the above assumptions and [54, Theorem 23.8], we have $\partial(f + h_1 + h_2) = \nabla f + \partial h_1 + \partial h_2$. Therefore, x^* is an optimal solution of (2.3.1) if and only if

$$0 \in \nabla f(x^*) + \partial h_1 (x^*) + \partial h_2 (x^*) . \tag{2.3.3}$$

Using Proposition 2.4(b), it then follows that x^* is an optimal solution of (2.3.1) if and only if there exists $y^* \in \mathbb{R}^n$ such that

$$0 \in \nabla f(x^*) + \partial h_1 (x^*) + y^*, \qquad 0 \in \partial h_2^* (y^*) - x^*. \tag{2.3.4}$$

It is interesting to note that the above inclusion problem is associated with the Lagrangian $\mathcal{L} : \mathcal{X} \times \mathcal{X} \to \bar{\mathbb{R}}$ defined as

$$\mathcal{L}(x, y) = f(x) + h_1(x) + \langle x, y\rangle - h_2^*(y), \quad \forall (x, y) \in \mathcal{X} \times \mathcal{X},$$

in that it can be simply expressed as

$$0 \in \partial_x \mathcal{L}(x, y), \quad 0 \in \partial_y(-\mathcal{L})(x, y), \tag{2.3.5}$$

where the two partial derivatives are with respect to the same inner product $\langle \cdot, \cdot \rangle$ on $\mathcal{X}$. Although one can apply the A-BD-HPE framework directly to the above system with $C = \partial(f + h_1)$ and $D = \partial h_2^*$, and $F(x, y) = (y, -x)$ for all $(x, y) \in \mathcal{X} \times \mathcal{X}$, it is more efficient from a computational point of view to introduce a scale factor to balance the two inclusions in (2.3.5).

Indeed, let $\theta > 0$ be given and consider the scaled inner product $\langle \cdot, \cdot \rangle_\theta$ in $\mathcal{X}$ defined as

$$\langle x, x' \rangle_\theta := \theta^{-1} \langle x, x' \rangle, \qquad \forall x, x' \in \mathcal{X},$$

and observe that the associated inner product norm, denoted by $\| \cdot \|_\theta$, satisfies

$$\| \cdot \|_\theta = \frac{1}{\sqrt{\theta}} \| \cdot \|. \tag{2.3.6}$$

Also, denote the gradient and ε-subdifferential of an arbitrary function $\phi : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ with respect to $\langle \cdot, \cdot \rangle_\theta$ by $\nabla^\theta \phi$ and $\partial_\varepsilon^\theta \phi$, respectively. It is trivial to see that

$$\nabla^\theta \phi = \theta(\nabla \phi), \qquad \partial_\varepsilon^\theta \phi = \theta(\partial_\varepsilon \phi). \tag{2.3.7}$$

It turns out that the monotone inclusion problem (2.3.5) is equivalent to

$$0 \in \partial_x^\theta \mathcal{L}(x,y), \quad 0 \in \partial_y\left(-\mathcal{L}(x,y)\right), \tag{2.3.8}$$

or equivalently,

$$\begin{aligned} 0 &\in \theta\big(\nabla f(x) + \partial h_1(x) + y\big), \\ 0 &\in \partial h_2^*(y) - x. \end{aligned} \tag{2.3.9}$$

We note that the use of (2.3.4), or more generally (2.3.9), as a way of solving (2.3.3) is well known (see for example the methods described in [11, 29, 39]).

The above system (2.3.8) is determined by $\mathcal{L}$ and the inner product norm on $\mathcal{X} \times \mathcal{X}$ defined as

$$\|(x,y)\|_{\theta,1} = \sqrt{\|x\|_\theta^2 + \|y\|^2}, \quad \forall (x,y) \in \mathcal{X} \times \mathcal{X}. \tag{2.3.10}$$

Note that this norm is the one given by (2.1.1) with $\mathcal{Z} = \mathcal{W} = \mathcal{X}$, $\|\cdot\|_{\mathcal{W}} = \|\cdot\|_\theta$ and $\|\cdot\|_{\mathcal{W}} = \|\cdot\|$. In order to view (2.3.8), or equivalently (2.3.9), as a special case of (2.1.1) and (2.1.2), the latter observation motivates us to define in this section $\mathcal{Z} := \mathcal{X}$, $\mathcal{W} := \mathcal{X}$, the inner products as

$$\langle \cdot,\cdot\rangle_{\mathcal{Z}} := \langle \cdot,\cdot\rangle_\theta, \qquad \langle \cdot,\cdot\rangle_{\mathcal{W}} := \langle \cdot,\cdot\rangle, \tag{2.3.11}$$

and the operators F, C, and D as

$$F(x,y) := (\theta y, -x),\ C(x) := \partial^\theta(f+h_1)(x) = \theta(\nabla f(x) + \partial h_1(x)),\ D(y) := \partial h_2^*(y), \tag{2.3.12}$$

for every $(x,y) \in \mathcal{X} \times \mathcal{X}$.

The following simple result summarizes the main properties of the scaled reformulation (2.3.9) (or equivalently, (2.3.8)) of (2.3.3).

Proposition 2.12. *The spaces $\mathcal{X}$ and $\mathcal{Y} := \mathcal{X}$ with corresponding inner products given by* (2.3.11)*, and the operators F, C and D defined by* (2.3.12)*, satisfy conditions A.1–A.4 with $L = \sqrt{\theta}$. Moreover, the inclusion problem* (2.3.9) *is equivalent to the maximal monotone inclusion problem* (2.1.2)*.*

Our goal now will be to state an instance of the A-BD-HPE framework for solving (2.3.9), and hence (2.3.1), under the assumption that the resolvents of both ∂h_1 and ∂h_2,

that is, the maps $(I+\lambda\partial h_i)^{-1}$ for every $\lambda>0$ and $i=1,2$, can be easily evaluated at any given $x\in\mathcal{X}$. In other words, we assume that the optimal solutions of minimization subproblems of the form

$$\min_{\bar{x}\in\mathcal{X}} h_i(x)+\frac{1}{2\lambda}\|\bar{x}-x\|^2$$

can be easily computed for any $x\in\mathcal{X}$, $\lambda>0$ and $i=1,2$.

Algorithm 2.2 Scaled A-BD-HPE method for (2.3.1)

0) Let $(x^0,y^0)\in\mathcal{X}\times\mathcal{X}$, $\theta>0$, $\sigma_1\in(0,1)$ and $\sigma\in[\sigma_1,1]$ be given, and set $k=1$ and

$$\tilde{\lambda}:=\min\left\{\frac{\sigma_1^2}{\theta L_f}\,,\,\frac{\sigma}{\sqrt{\theta}}\right\}; \tag{2.3.13}$$

1) set $\tilde{x}^k:=\left(I+\tilde{\lambda}\theta\partial h_1\right)^{-1}\left(x^{k-1}-\tilde{\lambda}\theta\left(\nabla f(x^{k-1})+y^{k-1}\right)\right)$;

2) set $\tilde{y}^k:=(I+\tilde{\lambda}\partial h_2^*)^{-1}(y^{k-1}+\tilde{\lambda}\tilde{x}^k)$;

3) choose λ_k to be the largest $\lambda>0$ such that

$$\|\lambda(v_1^k,v_2^k)+(\tilde{x}^k,\tilde{y}^k)-(x^{k-1},y^{k-1})\|_{\theta,1}^2+2\lambda\varepsilon_k\le\sigma^2\|(\tilde{x}^k,\tilde{y}^k)-(x^{k-1},y^{k-1})\|_{\theta,1}^2,$$

where

$$v_1^k:=\frac{1}{\tilde{\lambda}}\left(x^{k-1}-\tilde{x}^k\right)+\theta\left(\tilde{y}^k-y^{k-1}\right),\quad v_2^k:=\frac{1}{\tilde{\lambda}}(y^{k-1}-\tilde{y}^k),\quad \varepsilon_k:=\frac{L_f}{2}\|\tilde{x}^k-x^{k-1}\|^2; \tag{2.3.14}$$

4) set $(x^k,y^k)=(x^{k-1},y^{k-1})-\lambda_k(v_1^k,v_2^k)$ and $k\leftarrow k+1$, and go to step 1.

We now make two remarks about Algorithm 2.2. First, when $L_f=0$, it follows from (2.3.13) that $\tilde{\lambda}=\sigma/\sqrt{\theta}$ and hence that Algorithm 2.2 does not depend on the choice of σ_1. Second, the formula for computing $\tilde{y}^k$ in step 2 of Algorithm 2.2 involves the resolvent of ∂h_2^*, instead of ∂h_2. Using Lemma 2.1 with $T=\partial h_2^*$ and the fact that $(\partial h_2^*)^{-1}=\partial h_2$, it follows that $\tilde{y}^k$ can also be computed as

$$\tilde{y}^k=y^{k-1}+\tilde{\lambda}\tilde{x}^k-\tilde{\lambda}\left(I+\tilde{\lambda}^{-1}\partial h_2\right)^{-1}\left(\tilde{\lambda}^{-1}y^{k-1}+\tilde{x}^k\right). \tag{2.3.15}$$

Clearly, depending on the function h_2, one of these resolvents might be easier to compute than the other one, and hence is the better one for computing $\tilde{y}^k$. Using Lemma 2.1 with $T = \partial h_1$, it is also possible to give an expression for computing $\tilde{x}^k$ in terms of the resolvent of ∂h_1^*. Again, which one to use computationally will depend on the function h_1. We have chosen the formulae in steps 1 and 2 of Algorithm 2.2 due to their symmetry and the fact that they are more convenient for our theoretical presentation.

The following result shows that Algorithm 2.2 is a special instance of the A-BD-HPE framework applied to (2.3.8).

Lemma 2.13. *Consider the sequences $\{(x^k, y^k)\}$, $\{(\tilde{x}^k, \tilde{y}^k)\}$, $\{(v_1^k, v_2^k)\}$ and $\{\varepsilon_k\}$ generated by Algorithm 2.2 and, for every $k \in \mathbb{N}$, define*

$$z^k = x^k, \qquad w^k = y^k, \qquad \tilde{z}^k = \tilde{x}^k, \qquad \tilde{w}^k = \tilde{y}^k, \tag{2.3.16}$$

$$\tilde{\lambda}_k := \tilde{\lambda}, \qquad \varepsilon_k^z := \varepsilon_k, \qquad \varepsilon_k^w := 0, \tag{2.3.17}$$

and

$$\tilde{c}^k := \frac{x^{k-1} - \tilde{x}^k}{\tilde{\lambda}} - \theta y^{k-1}, \qquad \tilde{d}^k := \frac{y^{k-1} - \tilde{y}^k}{\tilde{\lambda}} + \tilde{x}^k. \tag{2.3.18}$$

Then, for every $k \in \mathbb{N}$, the following statements hold with respect to the A-BD-HPE framework with

$$\sigma_z = \sigma_1, \qquad \tilde{\sigma}_z = 0, \qquad \sigma_w = 0, \tag{2.3.19}$$

and the inner products $\langle \cdot, \cdot \rangle_{\mathcal{Z}}$ and $\langle \cdot, \cdot \rangle_{\mathcal{W}}$ and operators F, C and D defined as in (2.3.11) and (2.3.12):

a) $\tilde{\lambda}_k$ satisfies (2.1.3);

b) $\tilde{\lambda}_k$, z^{k-1} and the triple $(\tilde{z}^k, \tilde{c}^k, \varepsilon_k^z)$ satisfies (2.1.4) and (2.1.5) and

$$\theta^{-1}\tilde{c}^k \in \nabla f(x^{k-1}) + \partial h_1(\tilde{x}^k) \subseteq (\partial_{\varepsilon_k^z} f + \partial h_1)(\tilde{x}^k); \tag{2.3.20}$$

c) $\tilde{\lambda}_k$, w^{k-1} and the triple $(\tilde{w}^k, \tilde{d}^k, \varepsilon_k^w)$ satisfies (2.1.6), and

$$\tilde{d}^k \in \partial h_2^*(\tilde{y}^k); \tag{2.3.21}$$

d) $(v_1^k, v_2^k) = F(\tilde{z}^k, \tilde{w}^k) + (\tilde{c}^k, \tilde{d}^k)$.

As a consequence, Algorithm 2.2 applied to (2.3.1) *is a special instance of the A-BD-HPE framework for solving* (2.1.2) *where the inner products* $\langle \cdot, \cdot \rangle_{\mathcal{Z}}$ *and* $\langle \cdot, \cdot \rangle_{\mathcal{W}}$ *and operators* F, C *and* D *are given by* (2.3.11) *and* (2.3.12).

Proof. Statement a) follows immediately from condition (2.3.13), the definitions of σ_z, $\tilde{\sigma}_z$, σ_w and $\tilde{\lambda}_k$ in (2.3.17) and (2.3.19), and the fact that $L = \sqrt{\theta}$ and $\sigma_1 \leq \sigma$, in view of step 0 of Algorithm 2.2 and Proposition 2.12.

Now, it follows from (2.3.16), (2.3.18) and the definitions of F and $\tilde{\lambda}_k$ in (2.3.12) and (2.3.17), respectively, that

$$\tilde{\lambda}_k[F_1(\tilde{z}^k, w^{k-1}) + \tilde{c}^k] + \tilde{z}^k - z^{k-1} = \tilde{\lambda}[\theta y^{k-1} + \tilde{c}^k] + \tilde{x}^k - x^{k-1} = 0$$

and

$$\tilde{\lambda}_k[F_2(\tilde{z}^k, \tilde{w}^k) + \tilde{d}^k] + \tilde{w}^k - w^{k-1} = \tilde{\lambda}[-\tilde{x}^k + \tilde{d}^k] + \tilde{y}^k - y^{k-1} = 0.$$

Clearly, these identities and the definition of ε_k^w in (2.3.17) imply that $\tilde{\lambda}_k$, z^{k-1}, w^{k-1} and the triples $(\tilde{z}^k, \tilde{c}^k, \varepsilon_k^z)$ and $(\tilde{w}^k, \tilde{d}^k, \varepsilon_k^w)$ satisfy (2.1.5) and the inequality in (2.1.6). They also satisfy the inequality in (2.1.4) due to the fact that, the definitions of ε_k, $\tilde{\lambda}_k$, ε_k^z and σ_z in (2.3.14), (2.3.17) and (2.3.19), and relations (2.3.6), (2.3.13), (2.3.11) and (2.3.16), imply that

$$2\tilde{\lambda}_k \varepsilon_k^z = 2\tilde{\lambda}\varepsilon_k = L_f \tilde{\lambda}\|\tilde{x}^k - x^{k-1}\|^2 \leq \frac{\sigma_1^2}{\theta}\|\tilde{x}^k - x^{k-1}\|^2 = \sigma_z^2\|\tilde{x}^k - x^{k-1}\|_\theta^2 = \sigma_z^2\|\tilde{z}^k - z^{k-1}\|_{\mathcal{Z}}^2.$$

We will now show that the inclusions in (2.1.4) and (2.1.6) hold. It is well-known that Assumption B.2 implies that

$$f(\tilde{x}^k) - f(x^{k-1}) - \langle \nabla f(x^{k-1}), \tilde{x}^k - x^{k-1} \rangle \leq \frac{L_f}{2}\|\tilde{x}^k - x^{k-1}\|^2 = \varepsilon_k = \varepsilon_k^z,$$

where the last two equalities follow from the definitions of ε_k and ε_k^z in (2.3.14) and (2.3.17), respectively. Using the last conclusion, the fact that $\nabla f(x^{k-1}) \in \partial f(x^{k-1})$, Proposition 2.4(c) with $v = \nabla f(x^{k-1})$, $z = x^{k-1}$ and $\tilde{z} = \tilde{x}^k$, we then conclude that $\nabla f(x^{k-1}) \in \partial_{\varepsilon_k^z} f(\tilde{x}^k)$. Now, using the definition of $\tilde{x}^k$ in step 1 of Algorithm 2.2, $\tilde{c}^k$ in (2.3.18) and C

in (2.3.12), the last conclusion, relations (2.3.7) and (2.3.16), and Proposition 2.4(a), we conclude that

$$\begin{aligned}\tilde{c}^k &\in \theta[\nabla f(x^{k-1}) + \partial h_1(\tilde{x}^k)] \subseteq \theta(\partial_{\varepsilon_k^z} f + \partial h_1)(\tilde{x}^k) \subseteq \theta\left[\partial_{\varepsilon_k^z}(f+h_1)(\tilde{x}^k)\right] \\ &= \partial^\theta_{\varepsilon_k^z}(f+h_1)(\tilde{x}^k) \subseteq [\partial^\theta(f+h_1)]^{\varepsilon_k^z}(\tilde{x}^k) = C^{\varepsilon_k^z}(\tilde{x}^k) = C^{\varepsilon_k^z}(\tilde{z}^k),\end{aligned}$$

which shows that (2.3.20) and the inclusion in (2.1.4) hold. Also, the definitions of $\tilde{y}^k$ in step 2 of Algorithm 2.2, $\tilde{d}^k$ in (2.3.18), D in (2.3.12) and ε_k^w in (2.3.17), relation (2.3.16), and Proposition 2.2(d), imply that

$$\tilde{d}^k \in \partial h_2^*(\tilde{y}^k) = D(\tilde{y}^k) = D^{\varepsilon_k^w}(\tilde{y}^k) = D^{\varepsilon_k^w}(\tilde{w}^k),$$

which shows that (2.3.21) and the inclusion in (2.1.6) hold. We have thus shown statements b) and c).

Statement d) follows immediately from (2.3.16) and the definitions of F, v_1^k, v_2^k, $\tilde{c}^k$ and $\tilde{d}^k$ in (2.3.12), (2.3.14) and (2.3.18).

The last claim of the lemma immediately follows from (2.3.16), statements (a)–(d) and the descriptions of Algorithm 2.2 and the A-BD-HPE framework. □

It follows from Lemma 2.13 that Algorithm 2.2 is a special instance of the A-BD-HPE framework. Hence, the convergence result described in Theorem 2.7 applies to it. In what follows, we will describe the implications of this result towards the behavior of Algorithm 2.2.

However, we first make some observations regarding the distance of the initial point (x^0, y^0) to the solution set of (2.3.8) with respect to the norm $\|(\cdot,\cdot)\|_{\theta,1}$. First observe that the solution sets of (2.3.5) and (2.3.8) are the same. Second, by the saddle-point theory, this set is of the form $X^* \times Y^* \subseteq \mathcal{X} \times \mathcal{X}$. Third, the distance d_0^θ of the initial point (x^0, y^0) to the solution set of (2.3.8) with respect to the norm $\|(\cdot,\cdot)\|_{\theta,1}$ can be expressed as

$$d_0^\theta := \sqrt{\theta^{-1} d_{x,0}^2 + d_{y,0}^2}, \tag{2.3.22}$$

where

$$d_{x,0} := \min\{\|x - x^0\| : x \in X^*\}, \quad d_{y,0} := \min\{\|y - y^0\| : y \in Y^*\}. \tag{2.3.23}$$

The following theorem shows how Algorithm 2.2 nearly solves the optimality conditions (2.3.4) (or equivalently, (2.3.3)).

Theorem 2.14. *Consider the sequences* $\{(x^k, y^k)\}$, $\{(\tilde{x}^k, \tilde{y}^k)\}$, $\{(v_1^k, v_2^k)\}$ *and* $\{\varepsilon_k\}$ *generated by Algorithm 2.2 under the assumption that* $\sigma < 1$ *and, for every* $k \in \mathbb{N}$, *define*

$$r_1^k := \theta^{-1} v_1^k + \nabla f(\tilde{x}^k) - \nabla f(x^{k-1}), \tag{2.3.24}$$

$$r_2^k := v_2^k = \frac{1}{\tilde{\lambda}}(y^{k-1} - \tilde{y}^k). \tag{2.3.25}$$

Then, for every $k \in \mathbb{N}$,

$$r_1^k \in \nabla f(\tilde{x}^k) + \partial h_1(\tilde{x}^k) + \tilde{y}^k, \qquad r_2^k \in -\tilde{x}^k + \partial h_2^*(\tilde{y}^k), \tag{2.3.26}$$

and there exists $i \leq k$ *such that*

$$\sqrt{\theta\|r_1^i\|^2 + \|r_2^i\|^2} \leq \max\left\{\frac{1}{\sigma}, \frac{\sqrt{\theta}L_f}{\sigma_1^2}\right\}\left(\frac{1+\sigma+\sigma\min\left\{\sigma_1, \theta^{1/4}\sqrt{\sigma L_f}\right\}}{\sqrt{1-\sigma^2}}\right)\frac{\sqrt{\theta}}{\sqrt{k}}\sqrt{\theta^{-1}d_{x,0}^2 + d_{y,0}^2}.$$

Proof. Consider the sequences $\{\tilde{c}^k\}$ and $\{\tilde{d}^k\}$ defined in (2.3.18). It follows from the definition of v_1^k and v_2^k in (2.3.14) that

$$\theta^{-1} v_1^k = \tilde{y}^k + \theta^{-1}\tilde{c}^k, \qquad v_2^k = -\tilde{x}^k + \tilde{d}^k. \tag{2.3.27}$$

Hence, (2.3.26) follows from the above two identities, the first inclusion in (2.3.20), and relations (2.3.21), (2.3.24) and (2.3.25). Moreover, Lemma 2.13(d), the definitions of ε_k^z and ε_k^w in (2.3.17), together with Theorem 2.7 imply the existence of $i \leq k$ such that

$$\|(v_1^i, v_2^i)\|_{\theta,1} \leq d_0^\theta\sqrt{\frac{1+\sigma}{1-\sigma}\left(\frac{1}{\lambda_i\sum_{j=1}^k \lambda_j}\right)} \leq \sqrt{\frac{1+\sigma}{1-\sigma}}\frac{d_0^\theta}{\tilde{\lambda}\sqrt{k}}, \tag{2.3.28}$$

$$\varepsilon_i = \varepsilon_k^z + \varepsilon_k^w \leq \frac{\sigma^2\left(d_0^\theta\right)^2}{2(1-\sigma^2)\sum_{j=1}^k \lambda_j} \leq \frac{\sigma^2\left(d_0^\theta\right)^2}{2(1-\sigma^2)\tilde{\lambda}k}, \tag{2.3.29}$$

where the last inequalities in (2.3.28) and (2.3.29) follow from Proposition 2.6 and the definition of $\tilde{\lambda}_k$ in (2.3.17). Moreover, using the definitions of $\|\cdot\|_\theta$, $\|(\cdot,\cdot)\|_{\theta,1}$, ε_i, r_1^k and r_2^k in (2.3.6), (2.3.10), (2.3.14), (2.3.24) and (2.3.25), respectively, the triangular inequality

for norms and (2.3.2), we conclude that

$$\begin{aligned}
\sqrt{\theta\|r_1^i\|^2 + \|r_2^i\|^2} &= \|(\theta r_1^i, r_2^i)\|_{\theta,1} = \left\| \left(v_1^i + \theta[\nabla f(\tilde{x}^i) - \nabla f(x^{i-1})], v_2^i \right) \right\|_{\theta,1} \\
&\leq \|(v_1^i, v_2^i)\|_{\theta,1} + \theta \left\| \left(\nabla f(\tilde{x}^i) - \nabla f(x^{i-1}), 0 \right) \right\|_{\theta,1} \\
&= \|(v_1^i, v_2^i)\|_{\theta,1} + \sqrt{\theta}\|\nabla f(\tilde{x}^i) - \nabla f(x^{i-1})\| \\
&\leq \|(v_1^i, v_2^i)\|_{\theta,1} + \sqrt{\theta} L_f \|\tilde{x}^i - x^{i-1}\| \\
&= \|(v_1^i, v_2^i)\|_{\theta,1} + \sqrt{2\theta L_f \varepsilon_i}. \qquad (2.3.30)
\end{aligned}$$

Now, combining (2.3.28), (2.3.29) and (2.3.30), we have

$$\sqrt{\theta\|r_1^i\|^2 + \|r_2^i\|^2} \leq \left(\frac{1 + \sigma + \sigma\sqrt{\theta\tilde{\lambda} L_f}}{\sqrt{1-\sigma^2}} \right) \frac{d_0^\theta}{\tilde{\lambda}\sqrt{k}},$$

which, together with (2.3.22) and the definition of $\tilde{\lambda}$ in (2.3.13), imply the last conclusion of the theorem. □

Note that the point-wise iteration-complexity bound in Theorem 2.14 is $\mathcal{O}(1/\sqrt{k})$. Theorem A.3 derives an $\mathcal{O}(1/k)$ iteration-complexity ergodic bound for Algorithm 2.2 as an immediate consequence of Theorem 3.3 of [39] and Theorem 2.11(b).

2.3.2 Specialization of Algorithm 2.2 to conic optimization

In this section, we discuss the specialization of Algorithm 2.2 to the context of conic optimization problems possessing a two-easy-block structure.

More specifically, for $i = 1, 2$, let $\mathcal{W}_i$ be an inner product space whose inner product and associated norm is denoted by $\langle \cdot, \cdot \rangle_{\mathcal{W}_i}$ and $\|\cdot\|_{\mathcal{W}_i}$. We consider the conic optimization problem of the form

$$\begin{aligned}
z_P^* := \min\ & \langle c, x \rangle \\
\text{s.t. } & \mathcal{A}_1 x - b_1 \in \mathcal{K}_1 \qquad (2.3.31) \\
& \mathcal{A}_2 x - b_2 \in \mathcal{K}_2,
\end{aligned}$$

where $c \in \mathcal{X}$, $b_1 \in \mathcal{W}_1$, $b_2 \in \mathcal{W}_2$, $\mathcal{A}_1 : \mathcal{X} \to \mathcal{W}_1$ and $\mathcal{A}_2 : \mathcal{X} \to \mathcal{W}_2$ are linear maps, and $\mathcal{K}_1 \subseteq \mathcal{W}_1$ and $\mathcal{K}_2 \subseteq \mathcal{W}_2$ are nonempty closed convex cones. Observe that (2.3.31) is a

special of (2.3.1) in which

$$f(\cdot) = \langle c, \cdot \rangle, \qquad h_i(\cdot) = \delta_{\mathcal{M}_i}(\cdot) = \delta_{\mathcal{K}_i}(\mathcal{A}_i(\cdot) - b_i), \quad i = 1, 2, \tag{2.3.32}$$

and

$$\mathcal{M}_i := \{x \in \mathcal{X} : \mathcal{A}_i x - b_i \in \mathcal{K}_i\}, \qquad i = 1, 2. \tag{2.3.33}$$

Throughout this section, we make the following assumptions on (2.3.31):

E.1) (2.3.31) has an optimal solution, and hence $z_P^* \in \mathbb{R}$;

E.2) (2.3.31) has a Slater point, i.e., there exists $x \in \mathcal{X}$ such that $\mathcal{A}_i x - b_i \in \operatorname{ri} \mathcal{K}_i$ for $i = 1, 2$.

We will also need another assumption related to our ability to evaluate the resolvents $(I + \lambda \partial h_i)^{-1}$, $i = 1, 2$, at any given $x \in \mathcal{X}$. In the case of (2.3.31) with h_i defined as in (2.3.32), evaluating the resolvent of ∂h_i at x is equivalent to projecting x onto $\mathcal{M}_i$, i.e.,

$$(I + \lambda \partial h_i)^{-1}(x) = \Pi_{\mathcal{M}_i}(x) := \arg\min_{\tilde{x} \in \mathcal{X}} \left\{ \frac{1}{2} \|\tilde{x} - x\|^2 : \mathcal{A}_i \tilde{x} - b_i \in \mathcal{K}_i \right\}, \quad \forall \lambda > 0, \forall x \in \mathcal{X}. \tag{2.3.34}$$

Observe that $(I + \lambda \partial h_i)^{-1}(x)$ does not depend on the value of λ. The optimality conditions of the optimization problem above, Assumption E.2, the fact that $\delta_{\mathcal{M}_i}(\cdot) = \delta_{\mathcal{K}_i}(\mathcal{A}_i(\cdot) - b_i)$ and the well-known chain rule property of the subdifferential imply that p_i is the optimal solution of (2.3.34) if and only if $p_i \in \mathcal{M}_i$ and

$$p_i - x \in -\partial \delta_{\mathcal{M}_i}(p_i) = -\mathcal{A}_i^* \left[(\partial \delta_{\mathcal{K}_i}) (\mathcal{A}_i p_i - b_i) \right] = -\mathcal{A}_i^* N_{\mathcal{K}_i}(\mathcal{A}_i p_i - b_i),$$

where $\mathcal{A}_i^*$ is the adjoint of $\mathcal{A}_i$ and $N_{\mathcal{K}_i}$ denotes the normal cone operator for $\mathcal{K}_i$. Hence, in view of the characterization of the normal cone, we conclude that for every $\xi > 0$, $x \in \mathcal{X}$ and $i = 1, 2$, p_i is the optimal solution of (2.3.34) if and only if there exists a dual variable $w_i \in \mathcal{W}_i$ such that

$$\mathcal{A}_i p_i - b_i \in \mathcal{K}_i, \quad \mathcal{A}_i^* w_i = \xi(p_i - x), \quad w_i \in \mathcal{K}_i^*, \quad \langle w_i, \mathcal{A}_i p_i - b_i \rangle_{\mathcal{W}_i} = 0, \tag{2.3.35}$$

where $\mathcal{K}_i^*$ is the dual cone of $\mathcal{K}_i$. We further assume that:

E.3) for any given $\xi > 0$, $x \in \mathcal{X}$ and $i = 1, 2$, it is easy to compute a pair $(p_i, w_i) \in \mathcal{M}_i \times \mathcal{W}_i$ satisfying (2.3.35).

It should be observed that the application of Algorithm 2.2 to a conic programming instance strongly depends on the possibility of splitting its constraints into two blocks $\mathcal{M}_1$ and $\mathcal{M}_2$ as in (2.3.33) such that E.3, is satisfied. In this respect, the constraints of all instances used in the benchmarks of sections 2.3.4 and 2.3.5 can be partitioned so as to satisfy E.3, without the need of reformulating them. Another possibility of solving a general conic SDP instance (2.3.31) is to reformulate it in standard form (1.1.1) and apply Algorithm 2.2 with the partition given by the blocks $\mathcal{M}_1 = \mathcal{K}$ and $\mathcal{M}_2 = \mathcal{M} := \{x : \mathcal{A}x = b\}$. In fact, the latter approach is the one used by the DSA-BD method developed in [35].

The dual of (2.3.31) is the conic optimization problem given by

$$\begin{aligned} z_D^* := \max\ & \langle b_1, w_1 \rangle_{\mathcal{W}_1} + \langle b_2, w_2 \rangle_{\mathcal{W}_2} \\ \text{s.t. } & \mathcal{A}_1^* w_1 + \mathcal{A}_2^* w_2 = c, \\ & w_1 \in \mathcal{K}_1^*,\ w_2 \in \mathcal{K}_2^*. \end{aligned} \tag{2.3.36}$$

It is well-known that assumptions E.1 and E.2 imply that: i) the dual of (2.3.31) has an optimal solution and $z_P^* = z_D^*$; and ii) $x^* \in \mathcal{X}$ is an optimal solution of (2.3.31) and the pair $(w_1^*, w_2^*) \in \mathcal{W}_1 \times \mathcal{W}_2$ is an optimal solution of (2.3.36) if and only if

$$\begin{gathered} c - \mathcal{A}_1^* w_1^* - \mathcal{A}_2^* w_2^* = 0, \\ \mathcal{A}_i x^* - b_i \in \mathcal{K}_i, \quad w_i^* \in \mathcal{K}_i^*, \quad \langle w_i^*, \mathcal{A}_i x^* - b_i \rangle_{\mathcal{W}_i} = 0, \qquad i = 1, 2. \end{gathered}$$

For the sake of clarity we explicitly state below a specialization of Algorithm 2.2 to the context of (2.3.31), i.e., the special case of Algorithm 2.2 in which f, h_1 and h_2 are given by (2.3.32), $L_f = 0$ and the iterate $\tilde{y}^k$ in step 2 is computed using the alternative formula (2.3.15). In addition, steps 1 and 2 include the computation of a sequence of dual variables $\{(w_1^k, w_2^k)\} \subseteq \mathcal{K}_1^* \times \mathcal{K}_2^*$, which can be easily obtained in view of Assumption E.3.

Algorithm 2.3 Scaled A-BD-HPE method for (2.3.31)

0) Let $(x^0, y^0) \in \mathcal{X} \times \mathcal{X}$, $\theta > 0$ and $\sigma \in (0, 1]$ be given, and set $k = 1$ and $\tilde{\lambda} := \sigma/\sqrt{\theta}$;

1) set $\tilde{x}^k := \Pi_{\mathcal{M}_1}\left(x^{k-1} - \tilde{\lambda}\theta\left(c + y^{k-1}\right)\right)$, or equivalently, compute a pair $(\tilde{x}^k, w_1^k) \in \mathcal{X} \times \mathcal{W}_1$ such that

$$\mathcal{A}_1\tilde{x}^k - b_1 \in \mathcal{K}_1, \quad \mathcal{A}_1^* w_1^k = \frac{\tilde{x}^k - x^{k-1}}{\tilde{\lambda}\theta} + c + y^{k-1}, \quad w_1^k \in \mathcal{K}_1^*, \quad \langle w_1^k, \mathcal{A}_1\tilde{x}^k - b_1\rangle_{\mathcal{W}_1} = 0; \tag{2.3.37}$$

2) compute $\tilde{u}^k = \Pi_{\mathcal{M}_2}\left(\tilde{\lambda}^{-1}y^{k-1} + \tilde{x}^k\right)$, or equivalently, a pair $(\tilde{u}^k, w_2^k) \in \mathcal{X} \times \mathcal{W}_2$ satisfying

$$\mathcal{A}_2\tilde{u}^k - b_2 \in \mathcal{K}_2, \quad \mathcal{A}_2^* w_2^k = \tilde{\lambda}(\tilde{u}^k - \tilde{x}^k) - y^{k-1}, \quad w_2^k \in \mathcal{K}_2^*, \quad \langle w_2^k, \mathcal{A}_2\tilde{u}^k - b_2\rangle_{\mathcal{W}_2} = 0, \tag{2.3.38}$$

and set $\tilde{y}^k = y^{k-1} + \tilde{\lambda}(\tilde{x}^k - \tilde{u}^k)$;

3) choose λ_k to be the largest $\lambda > 0$ such that

$$\|\lambda(v_1^k, v_2^k) + (\tilde{x}^k, \tilde{y}^k) - (x^{k-1}, y^{k-1})\|_{\theta,1} \leq \sigma\|(\tilde{x}^k, \tilde{y}^k) - (x^{k-1}, y^{k-1})\|_{\theta,1}$$

where v_1^k and v_2^k are defined as in (2.3.14);

4) set $(x^k, y^k) = (x^{k-1}, y^{k-1}) - \lambda_k(v_1^k, v_2^k)$ and $k \leftarrow k + 1$, and go to step 1.

Observe that the condition in step 1 is equivalent to

$$\xi = (\tilde{\lambda}\theta)^{-1}, \qquad x = x^{k-1} - \tilde{\lambda}\theta(c + y^{k-1}), \qquad p_1 = \tilde{x}^k, \qquad w_1 = w_1^k$$

satisfying (2.3.35) with $i = 1$. Moreover, the condition in step 2 is equivalent to

$$\xi = \tilde{\lambda}, \qquad x = \tilde{\lambda}^{-1}y^{k-1} + \tilde{x}^k, \qquad p_2 = \tilde{u}^k = \tilde{x}^k + (y^{k-1} - \tilde{y}^k)/\tilde{\lambda}, \qquad w_2 = w_2^k$$

satisfying (2.3.35) with $i = 2$.

The following result specializes Theorem 2.14 to the context of (2.3.31) and also shows how Algorithm 2.3 solves the dual problem (2.3.36) in the limit.

Theorem 2.15. *Consider the sequences* $\{(\tilde{x}^k, \tilde{y}^k)\}$, $\{(x^k, y^k)\}$, $\{(w_1^k, w_2^k)\}$, $\{\tilde{u}^k\}$ *and* $\{(v_1^k, v_2^k)\}$ *generated by Algorithm 2.3 with* $\sigma < 1$ *and, for every* $k \in \mathbb{N}$, *define*

$$r^k := c - \mathcal{A}_1^* w_1^k - \mathcal{A}_2^* w_2^k. \tag{2.3.39}$$

Then, Algorithm 2.3 is a special case of Algorithm 2.2 with f, h_1 *and* h_2 *given by* (2.3.32) *and* $L_f = 0$. *Moreover, for every* $k \in \mathbb{N}$, *in addition to* (2.3.37) *and* (2.3.38), *the following statements hold:*

a) $v_1^k = \theta r^k$ *and* $v_2^k = \tilde{u}^k - \tilde{x}^k$;

b) the duality gap $dg_k := \langle c, \tilde{x}^k \rangle - \langle b_1, w_1^k \rangle_{\mathcal{W}_1} - \langle b_2, w_2^k \rangle_{\mathcal{W}_2}$ *can be alternatively computed as*

$$dg_k = \langle r^k, \tilde{x}^k \rangle + \langle v_2^k, \tilde{y}^k \rangle;$$

c) there exists $i \leq k$ *such that*

$$\max\left\{\sqrt{\theta}\|r^k\|\,,\, \|v_2^k\|\right\} \leq \frac{\sqrt{\theta}}{\sigma\sqrt{k}}\sqrt{\left(\frac{1+\sigma}{1-\sigma}\right)\left(\theta^{-1}d_{x,0}^2 + d_{y,0}^2\right)},$$

where $d_{x,0}$ *and* $d_{y,0}$ *are defined in* (2.3.23).

Proof. The first part of the theorem follows from (2.3.15), (2.3.32) and (2.3.34). To show a), note that the definition of $\tilde{y}^k$ in step 2 of Algorithm 2.3 and the definition of v_2^k in (2.3.14) imply that $v_2^k = \tilde{u}^k - \tilde{x}^k$. Moreover, in view of the definition of v_1^k in (2.3.14), the second relations in (2.3.37) and (2.3.38), and the definition of $\tilde{y}^k$ in step 2 of Algorithm 2.3, we have

$$\mathcal{A}_1^* w_1^k = c - \theta^{-1} v_1^k + \tilde{y}^k, \qquad \mathcal{A}_2^* w_2^k = -\tilde{y}^k. \tag{2.3.40}$$

The first identity in a) follows from the definition of r^k and the above two relations. To show b), note that the definition of r^k and the last relations in (2.3.37) and (2.3.38) imply that

$$\begin{aligned}\langle c, \tilde{x}^k \rangle - \langle b_1, w_1^k \rangle_{\mathcal{W}_1} + \langle b_2, w_2^k \rangle_{\mathcal{W}_2} &= \langle r^k + \mathcal{A}_1^* w_1^k + \mathcal{A}_2^* w_2^k, \tilde{x}^k \rangle - \langle \tilde{x}^k, \mathcal{A}_1^* w_1^k \rangle - \langle \tilde{u}^k, \mathcal{A}_2^* w_2^k \rangle \\ &= \langle r^k, \tilde{x}^k \rangle - \langle \tilde{u}^k - \tilde{x}^k, \mathcal{A}_2^* w_2^k \rangle = \langle r^k, \tilde{x}^k \rangle + \langle v_2^k, \tilde{y}^k \rangle,\end{aligned}$$

where the last equality follows from the second identities in (2.3.40) and a). Finally, c) follows from Theorem 2.14 with f, h_1 and h_2 given by (2.3.32) and $L_f = 0$, and the fact that r_1^k and r_2^k defined in (2.3.24) and (2.3.25) are equal to r^k and v_2^k, respectively, in view of a) and the fact that $\nabla f(\cdot) = c$. □

We now make some observations about Algorithm 2.3 and Theorem 2.15. First, Theorem 2.15 shows that $\tilde{x}^k$ and its perturbation $\tilde{u}^k = \tilde{x}^k + v_2^k$ satisfy the first and second blocks $\mathcal{A}_1 x - b_1 \in \mathcal{K}_1$ and $\mathcal{A}_2 x - b_2 \in \mathcal{K}_2$, respectively. Second, Algorithm 2.3 can be implemented without generating the dual sequence $\{(w_1^k, w_2^k)\}$. In such case, a) and b) of Theorem 2.15 show that the quantities $c - \mathcal{A}_1^* w_1^k - \mathcal{A}_2^* w_2^k$ and $\langle c, \tilde{x}^k \rangle - \langle b_1, w_1^k \rangle_{\mathcal{W}_1} - \langle b_2, w_2^k \rangle_{\mathcal{W}_2}$, commonly used to monitor the progress of algorithms for solving (2.3.31) and (2.3.36), can be computed in terms of $\tilde{x}^k$ and $\tilde{y}^k$ only, and hence do not require (w_1^k, w_2^k). Third, Theorem 2.15(c) sheds light on how the scaling parameter θ might affect the sizes of the residuals $r^k = c - \mathcal{A}_1^* w_1^k - \mathcal{A}_2^* w_2^k$ and $v_2^k = \tilde{u}^k - \tilde{x}^k$. Roughly speaking, viewing all quantities in the bound of Theorem 2.15(c), with the exception of θ, as constants, we see that

$$\|r^k\| = \mathcal{O}\left(\max\left\{1, \theta^{-1/2}\right\}\right), \qquad \|v_2^k\| = \mathcal{O}\left(\max\left\{1, \theta^{1/2}\right\}\right).$$

Hence, the ratio $\|v_2^k\|/\|r^k\|$ can grow significantly as $\theta \to \infty$, while it can become very small as $\theta \to 0$. This suggests that this ratio increases (resp., decreases) as θ increases (resp., decreases). In fact, we have observed in our computational experiments that this ratio behaves just as described.

2.3.3 A practical dynamically scaled BD method

In this subsection, we describe three measures that quantify the optimality of an approximate solution of (2.3.31), namely: the primal infeasibility measure; the dual infeasibility measure; and the relative duality gap. We also describe two important refinements of Algorithm 2.3, whose goal is to balance the magnitudes of the primal and dual infeasibility measures. More specifically, we describe: i) a scheme for choosing the initial scaling parameter θ; and ii) a procedure to dynamically update the scaling parameter θ for balancing the sizes of the primal and dual infeasibility measures as the algorithm progresses.

Let $\mathcal{X}$, $\mathcal{W}_1$ and $\mathcal{W}_2$ be as in Section 2.3.2. For the purpose of describing a stopping criterion for Algorithm 2.3, for $i = 1, 2$, denote the distance of a point $w \in \mathcal{W}_i$ to the cone $\mathcal{K}_i$ as

$$d_i(w) := \min\left\{\|w - \tilde{w}\|_{\mathcal{W}_i} : \tilde{w} \in \mathcal{K}_i\right\} \qquad \forall w \in \mathcal{W}_i,$$

and the primal infeasibility measure as

$$\epsilon_P(x) := \frac{\sqrt{d_1\left(\mathcal{A}_1 x - b_1\right)^2 + d_2\left(\mathcal{A}_2 x - b_2\right)^2}}{1 + \sqrt{\|b_1\|^2_{\mathcal{W}_1} + \|b_2\|^2_{\mathcal{W}_2}}} \qquad \forall x \in \mathcal{X}. \tag{2.3.41}$$

Also, define the dual infeasibility measure as

$$\epsilon_D(w_1, w_2) := \frac{\|c - \mathcal{A}_1^* w_1 - \mathcal{A}_2^* w_2\|}{\|c\| + 1} \qquad \forall (w_1, w_2) \in \mathcal{W}_1 \times \mathcal{W}_2. \tag{2.3.42}$$

Finally, define the relative duality gap as

$$\epsilon_G(x, w_1, w_2) := \frac{\langle c, x\rangle - \langle b_1, w_1\rangle_{\mathcal{W}_1} - \langle b_2, w_2\rangle_{\mathcal{W}_2}}{|\langle c, x\rangle| + |\langle b_1, w_1\rangle + \langle b_2, w_2\rangle| + 1} \qquad \forall x \in \mathcal{X},\ \forall (w_1, w_2) \in \mathcal{W}_1 \times \mathcal{W}_2. \tag{2.3.43}$$

For given tolerances $\bar{\epsilon}, \bar{\nu} > 0$, we stop Algorithm 2.3 whenever

$$\max\left\{\epsilon_{P,k}, \epsilon_{D,k}\right\} \le \bar{\epsilon}, \qquad |\epsilon_{G,k}| \le \bar{\nu}, \tag{2.3.44}$$

where

$$\epsilon_{P,k} := \epsilon_P(\tilde{x}^k), \qquad \epsilon_{D,k} := \epsilon_D(w_1^k, w_2^k), \qquad \epsilon_{G,k} := \epsilon_G(\tilde{x}^k, w_1^k, w_2^k).$$

We now make some observations about the stopping criteria (2.3.44). First, in view of Theorem 2.15, the first inclusion in (2.3.37) and the definition of r^k in (2.3.39), we have that

$$\epsilon_{P,k} = \frac{d_2\left(\mathcal{A}_2 \tilde{x}^k - b_2\right)}{1 + \sqrt{\|b_1\|^2_{\mathcal{W}_1} + \|b_2\|^2_{\mathcal{W}_2}}}, \quad \epsilon_{D,k} = \frac{\left\|r^k\right\|}{\|c\| + 1}, \quad \epsilon_{G,k} = \frac{\langle r^k, \tilde{x}^k\rangle + \langle v_2^k, \tilde{y}^k\rangle}{|\langle c, \tilde{x}^k\rangle| + \left|\langle r^k - c, \tilde{x}^k\rangle + \langle v_2^k, \tilde{y}^k\rangle\right| + 1}. \tag{2.3.45}$$

Second, since Theorem 2.15, (2.3.37) and (2.3.38) imply that zero is a cluster value of the sequences $\{\epsilon_{P,k}\}$, $\{\epsilon_{D,k}\}$ and $\{\epsilon_{G,k}\}$ as $k \to \infty$, Algorithm 2.3 with the termination criteria (2.3.44) will eventually terminate. Third, another possibility is to terminate Algorithm 2.3 based on the quantities $\epsilon'_{P,k} = \epsilon_P(\tilde{u}^k)$, $\epsilon_{D,k}$ and $\epsilon'_{G,k} := \epsilon_G(\tilde{u}^k, w_1^k, w_2^k)$, which also

approach zero (in a cluster sense) due to Theorem 2.15. Our current implementation of Algorithm 2.3 ignores the latter possibility and terminates based on (2.3.44). Fourth, the above termination criteria do not contain a violation measure with respect to the constraint $(w_1, w_2) \in \mathcal{K}_1^* \times \mathcal{K}_2^*$. In fact, our benchmarks of sections 2.3.4 and 2.3.5 disregard this measure due to the fact that all the codes tested generate the sequence $\{(w_1^k, w_2^k)\}$ inside the cone $\mathcal{K}_1^* \times \mathcal{K}_2^*$. Finally, Theorem 2.15(a) and the first inclusion in (2.3.38) imply that

$$\epsilon_{P,k} \leq \frac{\left\| \left(\mathcal{A}_2 \tilde{x}^k - b_2 \right) - \left(\mathcal{A}_2 \tilde{u}^k - b_2 \right) \right\|_{\mathcal{W}_2}}{1 + \sqrt{\|b_1\|_{\mathcal{W}_1}^2 + \|b_2\|_{\mathcal{W}_2}^2}} = \frac{\left\| \mathcal{A}_2 v_2^k \right\|_{\mathcal{W}_2}}{1 + \sqrt{\|b_1\|_{\mathcal{W}_1}^2 + \|b_2\|_{\mathcal{W}_2}^2}}. \tag{2.3.46}$$

We now discuss two important refinements of Algorithm 2.3 whose goal is to balance the magnitudes of the primal and dual infeasibility measures $\epsilon_{P,k}$ and $\epsilon_{D,k}$. First note that (2.3.45) and (2.3.46) imply that $\epsilon_{P,k}/\epsilon_{D,k} = \mathcal{O}\left(\|v_2^k\|/\|r^k\|\right)$. Hence, in view of the last observation in the paragraph following Theorem 2.15, the latter ratio can grow significantly as $\theta \to \infty$, while it can become very small as $\theta \to 0$. This suggests that this ratio increases (resp., decreases) as θ increases (resp., decreases). Indeed, our computational experiments indicate that the ratio $\epsilon_{P,k}/\epsilon_{D,k}$ behaves in this manner.

In the following, let θ_k denote the dynamic value of θ at the kth iteration of Algorithm 2.3. Observe that, in view of (2.3.45) and (2.3.14), the measures $\epsilon_{P,k}$ and $\epsilon_{D,k}$ depend on $\tilde{x}^k$ and $\tilde{y}^k$, whose values in turn depend on the choice of θ_k, in view of steps 1 and 2 of Algorithm 2.3. Hence, these two measures are indeed functions of θ, which are denoted as $\epsilon_{P,k}(\theta)$ and $\epsilon_{D,k}(\theta)$.

We first describe a scheme for choosing the initial scaling parameter θ_1. Let a constant $\rho > 1$ be given and set $\theta = 1$. If $\epsilon_{P,1}(\theta)/\epsilon_{D,1}(\theta) > \rho$ (resp., $\epsilon_{P,1}(\theta)/\epsilon_{D,1}(\theta) < \rho^{-1}$), we successively divide (resp., successively multiply) the current value of θ by 2 until $\epsilon_{P,1}(\theta)/\epsilon_{D,1}(\theta) \leq \rho$ (resp., $\epsilon_{P,1}(\theta)/\epsilon_{D,1}(\theta) \geq \rho^{-1}$) is satisfied, and set $\theta_1 = \theta_1^*$, where θ_1^* is the last value of θ. Since there is no guarantee that this procedure will terminate, we specify an upper bound on the number of times that θ can be updated. In our implementation, we set this upper bound to be 20.

We next describe a procedure for dynamically updating the scaling parameter θ so as to balance the sizes of the two measures $\epsilon_{P,k}(\theta)$ and $\epsilon_{D,k}(\theta)$. As in [35], we use the heuristic

of changing θ every time a specified number of iterations have been performed. More specifically, given an integer $\bar{k} \geq 1$, and scalars $\gamma > 1$ and $0 < \tau < 1$, we use the following dynamic scaling procedure for updating θ_k,

$$\theta_k = \begin{cases} \theta_{k-1}, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma^{-1} \leq \bar{\epsilon}_{P,k-1}/\bar{\epsilon}_{D,k-1} \leq \gamma \\ \tau^2\theta_{k-1}, & k \equiv 0 \bmod \bar{k} \text{ and } \bar{\epsilon}_{P,k-1}/\bar{\epsilon}_{D,k-1} > \gamma \\ \tau^{-2}\theta_{k-1}, & k \equiv 0 \bmod \bar{k} \text{ and } \bar{\epsilon}_{P,k-1}/\bar{\epsilon}_{D,k-1} < \gamma^{-1} \end{cases} \qquad \forall k \geq 2, \tag{2.3.47}$$

where

$$\bar{\epsilon}_{P,k-1} = \left(\prod_{i=k-\bar{k}}^{k-1} \epsilon_{P,i} \right)^{1/\bar{k}}, \qquad \bar{\epsilon}_{D,k-1} = \left(\prod_{i=k-\bar{k}}^{k-1} \epsilon_{D,i} \right)^{1/\bar{k}} \qquad \forall k > \bar{k}. \tag{2.3.48}$$

Roughly speaking, the above dynamic scaling procedure changes the value of θ at most a single time in the right direction so as to balance the sizes of the residuals based on the information provided by their values at the previous $\bar{k}$ iterations. We observe that a dynamic scaling procedure using $\epsilon_{P,k-1}$ and $\epsilon_{D,k-1}$ in place of $\bar{\epsilon}_{P,k-1}$ and $\bar{\epsilon}_{D,k-1}$ in (2.3.47), respectively, is proposed in Section 2.2. However, the more conservative procedure based on the aggregated measures in (2.3.48) have performed better in our computational experiments.

In our computational experiments, we will refer to the variant of Algorithm 2.3 which incorporates the two aforementioned schemes as the *two-easy-block-decomposition HPE* (2EBD-HPE) method. To illustrate the importance of the above two schemes, we have chosen an instance of a conic optimization problem to compare the performance of 2EBD-HPE against the performance of its two variants obtained by removing exactly one of the two schemes. Indeed, Figure 2.3.1 compares the performance of 2EBD-HPE against its variant VAR1 in which θ_1 is initialized as 1 instead of θ_1^*. Figure 2.3.2 compares the performance of 2EBD-HPE against its variant VAR2 in which dynamic scaling is removed (i.e., θ_k set to θ_1^* , for every $k \geq 1$).

In addition, to illustrate the importance of adaptively choosing the stepsize λ_k in Algorithm 2.3, Figure 2.3.3 compares the performance of 2EBD-HPE against its variant VAR3 in which the stepsize λ_k is chosen as $\tilde{\lambda} = \sigma/\sqrt{\theta_k}$ for every $k \geq 1$.

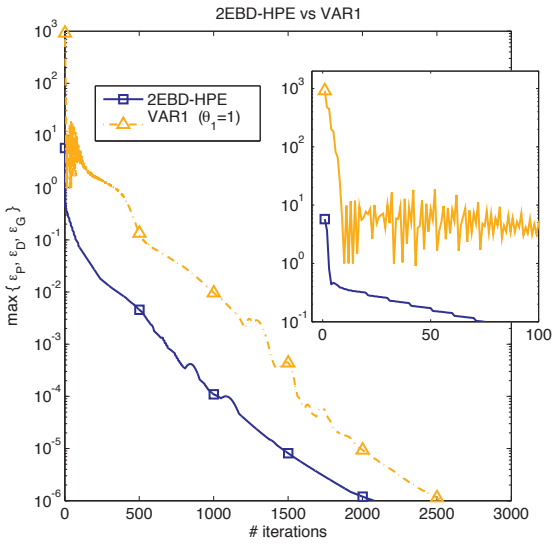

Figure 2.3.1: This example (BIQ-be200.8.8) illustrates how the scheme for choosing the initial scaling parameter θ_1 can help Algorithm 2.3 to start with an error at least 2 orders of magnitude smaller.

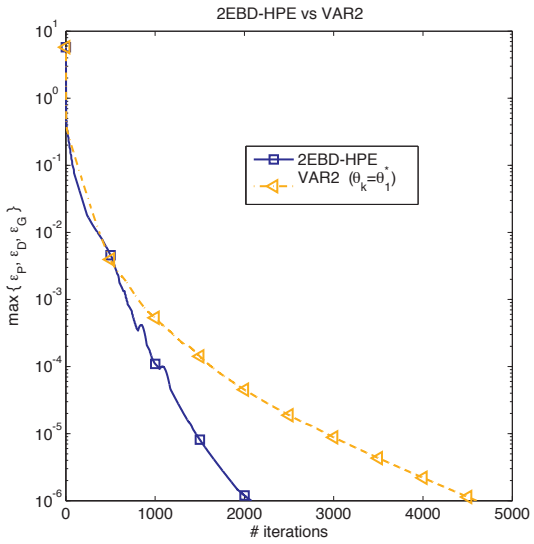

Figure 2.3.2: This example (BIQ-be200.8.8) illustrates how the dynamic scaling improves the performance of Algorithm 2.2 considerably.

Finally, Figure 2.3.4 compares the performance of 2EBD-HPE against the following three variants: i) VAR2, namely, the one that removes the dynamic scaling (i.e., set $\theta_k = \theta_1^*$, for every $k \geq 1$); ii) VAR4, namely, the one that removes the dynamic scaling and the initialization scheme for θ_1 (i.e., set $\theta_k = 1$, for every $k \geq 1$); and iii) VAR5, namely, the one that removes these latter two refinements and the use of adaptive stepsize (i.e., set $\theta_k = 1$ and $\lambda_k = \tilde{\lambda} = \sigma$, for every $k \geq 1$).

2.3.4 Numerical results: part I

In this subsection, we compare the 2EBD-HPE method described in Section 2.3.3 with a variant of the boundary point method, namely SDPAD, presented in [67]. More specifically, we compare these two methods on four important classes of graph-related SDP problems, namely: SDP relaxations of binary integer quadratic (BIQ) and frequency assignment (FAP) problems, and SDPs for θ-functions and θ_+-functions of graph stable set problems. This section contains three subsections. The first subsection considers SDP relaxations of BIQ

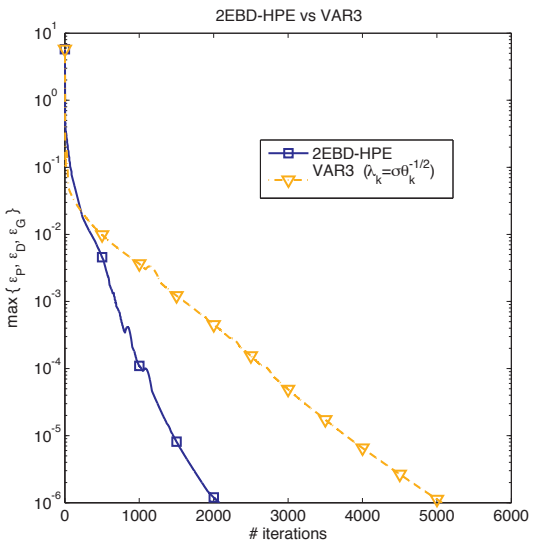

Figure 2.3.3: This example (BIQ-be200.8.8) illustrates how the adaptive stepsize improves the performance of Algorithm 2.2 considerably.

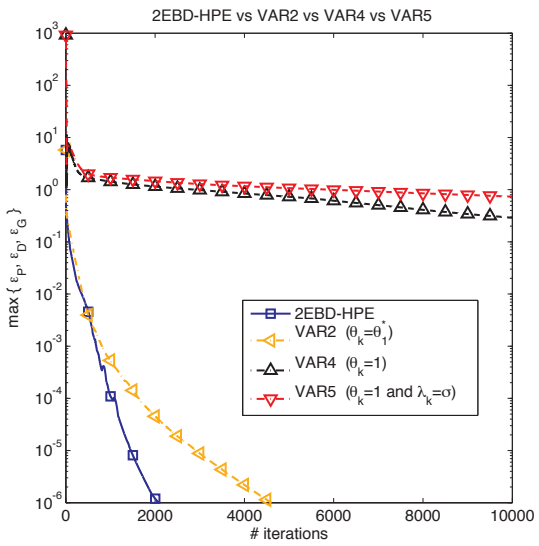

Figure 2.3.4: This example (BIQ-be200.8.8) illustrates how all the refinements made in the application of the BD-HPE framework to conic optimization helped improve the performance of the algorithm.

problems, the second one deals with SDP relaxations of FAPs, and the third one discusses SDPs corresponding to the θ-functions and θ_+-functions of graph stable set problems.

For the four problem classes above, $\mathcal{X}$, $\mathcal{W}_1$ and $\mathcal{W}_2$ are Cartesian products of Euclidean spaces and/or spaces of symmetric matrices, which are endowed with the natural canonical inner product consisting of the sum of Euclidean and/or Frobenius inner products associated with the spaces comprising the products.

We have implemented 2EBD-HPE for solving (2.3.31) in a MATLAB code. We have used the beta2 MATLAB implementation of SDPAD[1] released on December, 2012. The computational results for all the conic SDP instances were obtained on a single core of a server with 2 Xeon X5520 processors at 2.27GHz and 48GB RAM. For each one of the above classes of conic SDP problems, both methods generate primal and dual sequences $\{\tilde{x}^k\}$ and $\{(w_1^k, w_2^k)\} \subseteq \mathcal{K}_1^* \times \mathcal{K}_1^*$, and stop whenever (2.3.44) with $\bar{\epsilon} = 10^{-6}$ and $\bar{\nu} = 10^{-5}$ is satisfied.

[1]Available at `http://math.sjtu.edu.cn/faculty/zw2109/code/SDPAD-release-beta2.zip`.

For all classes of conic SDP problems considered the sequence $\{\tilde{x}^k\}$ lies in $\mathcal{S}^n$ for some $n \geq 1$, and evaluation of $\epsilon_{P,k}$ requires the computation of the distance from $\tilde{x}^k$ to $\mathcal{S}^n_+$, which in turn requires an eigenvalue decomposition of $\tilde{x}^k$. The 2EBD-HPE method has the nice feature that it generates $\{\tilde{x}^k\}$ inside $\mathcal{S}^n_+$. On the other hand, we have observed that SDPAD may generate elements of the sequence $\{\tilde{x}^k\}$ outside $\mathcal{S}^n_+$, but that this sequence eventually approaches $\mathcal{S}^n_+$ as $k \to \infty$ (as proved in Subsection 3.3 of [67]). For the purpose of this benchmark, we have assumed that SDPAD generates $\tilde{x}^k$ inside $\mathcal{S}^n_+$ so as to avoid computing an extra eigenvalue decomposition in the evaluation $\epsilon_{P,k}$ at every iteration.

We now make some general remarks about how the results are reported on the tables given below. Tables 2.3.1, 2.3.3, 2.3.5 and 2.3.7 compare 2EBD-HPE against SDPAD (each one of these tables corresponds to one of the four problem classes considered). The time (in seconds) taken by any of the two methods for any particular instance is marked in red, and also with an asterisk (*), whenever it cannot solve the instance by the required accuracy, in which case the residual (i.e., the maximum between the infeasibility measures and the relative duality gap) reported is the one obtained at the last iteration of the method. Also, the times marked in blue in a row is the best one among the ones listed in that row with the convention that, when a method cannot solve the instance, the corresponding time is assumed to be ∞. Tables 2.3.2, 2.3.4, 2.3.6 and 2.3.8 report more detailed computational results obtained at the last iteration of 2EBD-HPE, such as the primal and dual objective function values, number of iterations, the primal and dual infeasibility measures and the relative duality gaps as described in (2.3.41), (2.3.42) and (2.3.43), respectively (each one of these tables corresponds to one of the four problem classes considered).

Finally, Figures 2.3.5, 2.3.6, 2.3.7 and 2.3.8 plot the performance profiles (see [15]) of 2EBD-HPE and SDPAD methods for each of the four problem classes.

2.3.4.1 Binary integer quadratic problems

This subsection gives more details of our implementation of 2EBD-HPE for solving SDP relaxations of BIQ problems and summarizes its computational performance against SDPAD on a collection of 134 such instances. Recall that the SDP relaxation of the BIQ problem

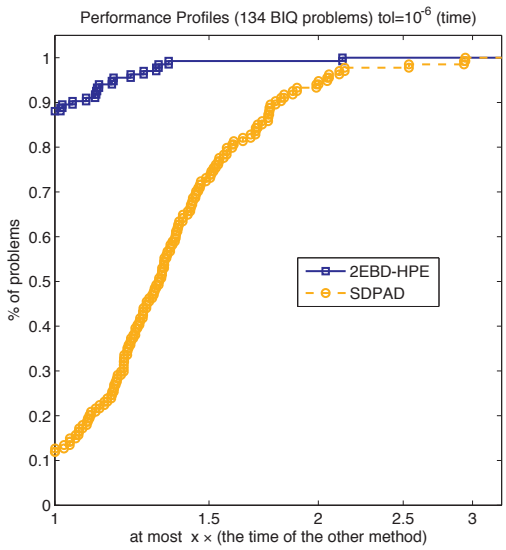

Figure 2.3.5: Performance profiles of 2EBD-HPE and SDPAD for solving 134 SDP relaxations of BIQ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

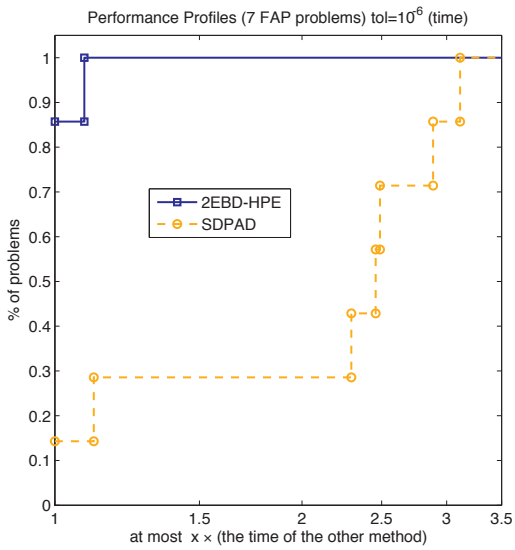

Figure 2.3.6: Performance profiles of 2EBD-HPE and SDPAD for solving 7 SDP relaxations of FAPs with accuracy $\bar{\epsilon} = 10^{-6}$.

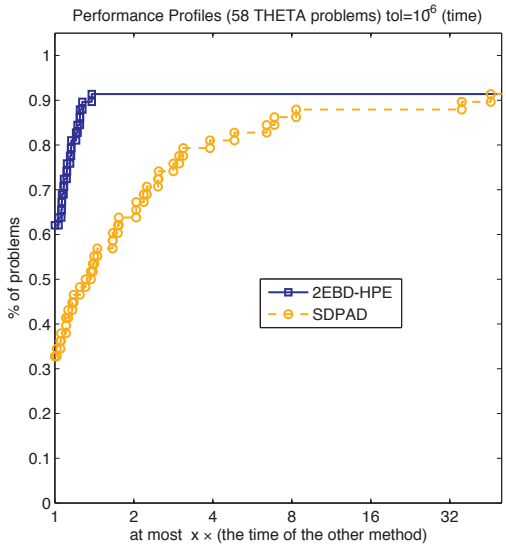

Figure 2.3.7: Performance profiles of 2EBD-HPE and SDPAD for solving 58 $\theta(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

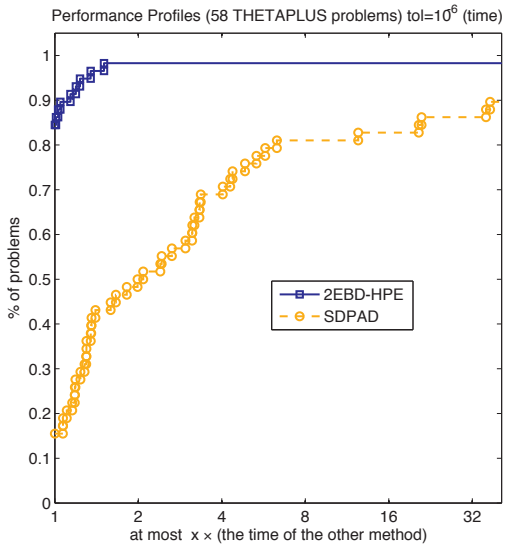

Figure 2.3.8: Performance profiles of 2EBD-HPE and SDPAD for solving 58 $\theta_+(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

is described in (2.2.26) in Subsection 2.2.2.3.

There is more than one way of viewing (2.2.26) as a special case of the two-easy-block structure formulation (2.3.31). For our current implementation, we have used the following formulation. Let $\mathcal{X} = \mathcal{W}_1 := \mathcal{S}^{n+1}$, $\mathcal{W}_2 = \mathbb{R}^n \times \mathbb{R} \times \mathcal{S}^n \times \mathbb{R}^n$, $\mathcal{K}_1 = \mathcal{S}^{n+1}_+$ and $\mathcal{K}_2 = \mathbf{0}_n \times \mathbf{0}_1 \times \mathbb{R}^{n(n+1)/2}_+ \times \mathbb{R}^n_+$, where $\mathbf{0}_n$ denotes an n dimensional vector of all zeros. With these definitions, we can easily see that (2.2.26) can be viewed as having the two-easy-block structure (2.3.31) if (2.2.26a) is chosen as $\mathcal{M}_1$ and (2.2.26b) are chosen as $\mathcal{M}_2$. Note that, in view of the first inclusion in (2.3.37), the constraint $x \succeq 0$ is always satisfied by 2EBD-HPE, while SDPAD approaches it in the limit.

Table 2.3.1 compares the two methods on a collection of 134 SDP relaxations of BIQ problems. For the purpose of this comparison, we have run 2EBD-HPE with $\sigma = 0.99$ and the values of γ, τ and $\bar{k}$ in the dynamic scaling rule (2.3.47) set to $\gamma = 1.5$, $\tau = 0.9$ and $\bar{k} = 10$. Table 2.3.2 gives more detailed computational results obtained by 2EBD-HPE (see the second paragraph preceding Subsection 2.3.4.1 for an explanation on this table). Figure 2.3.5 plots the performance profiles of both methods on this collection of 134 SDP relaxations of BIQ problems.

Note that 2EBD-HPE solves 119 (out of a total of 134) problems faster than SDPAD. Moreover, 2EBD-HPE solves about 9 of them at least 2 times faster than SDPAD.

Table 2.3.1: Comparison of the methods on BIQ problems

Problem		$\max\{\epsilon_P, \epsilon_D\}$		ϵ_G		**Time**	
Instance	$n_s \vert m$	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**
be100.1	101—5252	9.69 -7	9.96 -7	-1.17 -6	-2.38 -7	8.1	10.2
be100.10	101—5252	9.99 -7	1.00 -6	+5.75 -7	-4.82 -7	7.1	8.6
be100.2	101—5252	1.00 -6	9.98 -7	+1.23 -7	+3.44 -9	6.9	13.7
be100.3	101—5252	9.98 -7	9.99 -7	+2.53 -7	-1.33 -7	9.6	11.7
be100.4	101—5252	9.98 -7	9.99 -7	-1.32 -7	-2.86 -7	10.4	19.7
be100.5	101—5252	1.00 -6	9.99 -7	+2.93 -7	-6.37 -7	7.2	10.7
be100.6	101—5252	1.00 -6	9.98 -7	-5.03 -7	-5.47 -7	8.1	13.8
be100.7	101—5252	9.91 -7	9.99 -7	-1.08 -6	-1.40 -7	7.5	11.8
be100.8	101—5252	9.95 -7	9.89 -7	-9.80 -7	+5.54 -7	7.3	8.7
be100.9	101—5252	9.95 -7	9.99 -7	-2.56 -7	-2.33 -7	7.4	13.0
be120.3.1	121—7502	9.99 -7	9.97 -7	-2.27 -7	-3.99 -7	12.3	18.6
be120.3.10	121—7502	9.96 -7	1.00 -6	-4.58 -7	-6.97 -7	9.8	16.6
be120.3.2	121—7502	9.98 -7	9.99 -7	-6.31 -7	-1.08 -7	13.6	23.8
be120.3.3	121—7502	9.96 -7	9.98 -7	-5.51 -7	-8.88 -7	11.2	14.3
be120.3.4	121—7502	9.95 -7	9.98 -7	-1.23 -6	-2.02 -6	12.2	14.6
be120.3.5	121—7502	1.00 -6	1.00 -6	+1.91 -9	+2.25 -8	27.1	35.1
be120.3.6	121—7502	9.98 -7	9.98 -7	-4.74 -7	-2.67 -7	15.4	21.6
be120.3.7	121—7502	1.00 -6	9.99 -7	-1.33 -7	-1.75 -7	27.6	46.1
be120.3.8	121—7502	1.00 -6	1.00 -6	-2.17 -7	-1.22 -7	20.1	32.1
be120.3.9	121—7502	1.00 -6	1.00 -6	-2.58 -7	-4.54 -7	18.5	54.2
be120.8.1	121—7502	9.95 -7	9.96 -7	+7.08 -7	-2.38 -7	8.9	13.1
be120.8.10	121—7502	9.97 -7	1.00 -6	-1.87 -7	-4.31 -7	10.7	19.6
be120.8.2	121—7502	1.00 -6	9.99 -7	-1.87 -7	-1.71 -7	19.8	35.3
be120.8.3	121—7502	9.96 -7	9.98 -7	-3.19 -8	-3.89 -7	11.5	15.5
be120.8.4	121—7502	9.98 -7	1.00 -6	-2.02 -7	-4.79 -8	14.2	24.1
be120.8.5	121—7502	9.96 -7	9.99 -7	-3.26 -8	-2.07 -8	12.1	21.2
be120.8.6	121—7502	1.00 -6	9.97 -7	-1.66 -7	-3.36 -7	11.6	21.0
be120.8.7	121—7502	9.97 -7	9.94 -7	+3.12 -8	-4.51 -7	11.7	11.5
be120.8.8	121—7502	9.98 -7	1.00 -6	-3.22 -7	-4.03 -7	9.4	11.7
be120.8.9	121—7502	9.99 -7	9.97 -7	+3.53 -9	-2.13 -7	10.0	12.6
be150.3.1	151—11627	9.97 -7	9.98 -7	-8.52 -7	-6.45 -7	22.3	26.0
be150.3.10	151—11627	9.99 -7	9.99 -7	-3.00 -7	-1.34 -7	30.0	63.5
be150.3.2	151—11627	9.98 -7	9.98 -7	-5.18 -7	-4.05 -7	21.4	33.7
be150.3.3	151—11627	9.95 -7	1.00 -6	-6.71 -7	-3.33 -7	18.8	25.0
be150.3.4	151—11627	9.97 -7	9.98 -7	-5.55 -7	-2.94 -7	22.3	32.4

be150.3.5	151—11627	1.00 -6	9.94 -7	-1.33 -8	-6.97 -8	27.6	29.7
be150.3.6	151—11627	9.98 -7	1.00 -6	-4.99 -7	-1.06 -7	20.3	29.1
be150.3.7	151—11627	9.98 -7	9.97 -7	-4.94 -7	-5.11 -7	21.2	29.4
be150.3.8	151—11627	9.99 -7	1.00 -6	-1.01 -7	-1.56 -7	26.6	34.1
be150.3.9	151—11627	9.96 -7	9.98 -7	-7.71 -7	-8.63 -7	12.3	18.8
be150.8.1	151—11627	9.95 -7	9.99 -7	-2.49 -7	+2.11 -7	15.9	20.3
be150.8.10	151—11627	9.99 -7	9.99 -7	-2.07 -7	-1.79 -7	22.2	30.5
be150.8.2	151—11627	9.98 -7	9.98 -7	-9.87 -7	-4.89 -7	16.5	23.6
be150.8.3	151—11627	9.98 -7	9.98 -7	-1.91 -7	-4.82 -7	19.9	26.7
be150.8.4	151—11627	9.98 -7	9.99 -7	-3.01 -7	-2.38 -7	20.3	37.9
be150.8.5	151—11627	9.95 -7	1.00 -6	-5.86 -7	-4.96 -7	22.3	30.7
be150.8.6	151—11627	9.61 -7	1.00 -6	-1.52 -7	-1.70 -8	26.8	47.1
be150.8.7	151—11627	9.98 -7	9.99 -7	-8.41 -7	-2.00 -7	28.3	44.1
be150.8.8	151—11627	9.95 -7	1.00 -6	-4.89 -7	-3.57 -7	27.1	41.6
be150.8.9	151—11627	9.99 -7	1.00 -6	-2.79 -7	-3.28 -7	30.6	44.6
be200.3.1	201—20502	9.97 -7	9.99 -7	-9.07 -7	-8.88 -7	41.2	43.5
be200.3.10	201—20502	9.99 -7	1.00 -6	-2.37 -7	-2.61 -7	45.6	54.7
be200.3.2	201—20502	9.97 -7	9.98 -7	-5.55 -7	-5.00 -7	43.1	53.6
be200.3.3	201—20502	1.00 -6	1.00 -6	-2.95 -7	-3.61 -7	65.8	100.0
be200.3.4	201—20502	9.98 -7	9.99 -7	-6.64 -7	-2.99 -7	43.7	52.8
be200.3.5	201—20502	9.99 -7	9.99 -7	-5.50 -7	-2.38 -7	60.5	78.9
be200.3.6	201—20502	1.00 -6	9.99 -7	-4.07 -7	-8.00 -8	33.4	44.2
be200.3.7	201—20502	9.99 -7	9.97 -7	-3.89 -7	-2.44 -7	53.3	58.4
be200.3.8	201—20502	9.97 -7	9.99 -7	-7.34 -7	+5.36 -8	41.8	56.6
be200.3.9	201—20502	9.99 -7	9.99 -7	-9.61 -7	-8.34 -7	74.1	93.4
be200.8.1	201—20502	9.98 -7	1.00 -6	-6.96 -7	-3.49 -7	58.5	67.2
be200.8.10	201—20502	9.99 -7	9.99 -7	-5.35 -7	-3.54 -7	43.0	61.5
be200.8.2	201—20502	9.97 -7	9.98 -7	-5.82 -7	-4.65 -7	35.4	41.7
be200.8.3	201—20502	9.99 -7	1.00 -6	-5.89 -7	-3.40 -7	48.9	64.8
be200.8.4	201—20502	9.99 -7	9.89 -7	-6.81 -7	-6.44 -7	41.0	51.3
be200.8.5	201—20502	9.99 -7	9.99 -7	-1.45 -7	-1.92 -7	41.6	54.7
be200.8.6	201—20502	9.98 -7	1.00 -6	-1.15 -7	-1.53 -7	60.3	85.5
be200.8.7	201—20502	9.99 -7	1.00 -6	-2.26 -6	-2.48 -6	39.9	53.7
be200.8.8	201—20502	9.99 -7	9.98 -7	-4.88 -9	-9.53 -9	55.2	60.2
be200.8.9	201—20502	9.96 -7	9.98 -7	-2.23 -7	-1.76 -7	57.3	68.8
be250.1	251—31877	9.99 -7	1.00 -6	-2.39 -7	-1.29 -7	128.4	171.1
be250.10	251—31877	1.00 -6	1.00 -6	-3.26 -7	-2.49 -7	168.8	232.0
be250.2	251—31877	9.99 -7	1.00 -6	-8.47 -7	-4.63 -7	107.8	129.3
be250.3	251—31877	1.00 -6	1.00 -6	-1.95 -7	-1.81 -7	104.8	115.2
be250.4	251—31877	9.96 -7	1.00 -6	-6.92 -7	-1.01 -6	210.2	239.4

be250.5	251—31877	1.00 -6	1.00 -6	-5.84 -7	-2.47 -7	121.7	147.8
be250.6	251—31877	9.98 -7	9.99 -7	-5.34 -7	-4.31 -7	92.6	113.0
be250.7	251—31877	1.00 -6	9.99 -7	-1.39 -7	-1.75 -8	117.7	153.0
be250.8	251—31877	9.99 -7	1.00 -6	-1.90 -7	-1.67 -7	109.5	129.2
be250.9	251—31877	9.90 -7	9.99 -7	-3.96 -7	-2.72 -7	146.1	169.4
bqp100-1	101—5252	9.99 -7	9.96 -7	-2.66 -7	-4.23 -7	7.9	11.4
bqp100-10	101—5252	1.00 -6	9.92 -7	-1.95 -7	-3.28 -7	24.6	31.7
bqp100-2	101—5252	9.98 -7	1.00 -6	-4.71 -7	-5.04 -7	16.5	21.7
bqp100-3	101—5252	1.00 -6	1.00 -6	-6.10 -8	-2.17 -7	23.1	37.1
bqp100-4	101—5252	9.99 -7	9.99 -7	+9.60 -8	-1.44 -7	13.6	29.1
bqp100-5	101—5252	1.00 -6	9.99 -7	-7.86 -8	-2.19 -7	19.1	28.8
bqp100-6	101—5252	9.96 -7	9.99 -7	+8.45 -7	-2.50 -7	7.2	10.0
bqp100-7	101—5252	9.95 -7	9.95 -7	-6.41 -7	-7.52 -7	10.1	13.5
bqp100-8	101—5252	1.00 -6	1.00 -6	-8.85 -8	-3.52 -7	16.9	27.7
bqp100-9	101—5252	9.99 -7	9.99 -7	+7.87 -9	+2.75 -8	15.9	23.3
bqp250-1	251—31877	9.99 -7	1.00 -6	-4.68 -7	-1.85 -7	119.2	138.6
bqp250-10	251—31877	9.98 -7	9.99 -7	-1.09 -6	-1.24 -6	84.0	89.4
bqp250-2	251—31877	9.99 -7	9.99 -7	-7.66 -7	-7.04 -7	119.9	119.9
bqp250-3	251—31877	9.99 -7	9.96 -7	-1.75 -6	-4.02 -7	116.4	126.6
bqp250-4	251—31877	9.99 -7	9.92 -7	-8.32 -7	-1.25 -7	79.6	108.5
bqp250-5	251—31877	1.00 -6	9.99 -7	-8.75 -7	-4.82 -7	166.5	230.2
bqp250-6	251—31877	9.99 -7	9.99 -7	-5.26 -7	-6.67 -7	128.0	153.5
bqp250-7	251—31877	1.00 -6	9.99 -7	-8.46 -7	-6.95 -7	107.4	132.7
bqp250-8	251—31877	9.98 -7	9.99 -7	-4.61 -7	-6.12 -7	95.9	85.3
bqp250-9	251—31877	9.99 -7	9.99 -7	-4.00 -7	-2.11 -7	118.5	162.5
bqp500-1	501—126252	9.99 -7	9.99 -7	-1.32 -6	-3.06 -7	992.8	755.9
bqp500-10	501—126252	9.99 -7	9.93 -7	-1.47 -6	-1.32 -6	1042.5	932.9
bqp500-2	501—126252	9.98 -7	9.99 -7	-9.21 -7	-1.01 -7	1113.0	1140.2
bqp500-3	501—126252	1.00 -6	9.99 -7	-1.47 -6	+3.48 -7	1032.3	925.9
bqp500-4	501—126252	1.00 -6	9.97 -7	-1.24 -6	-3.88 -7	970.1	926.0
bqp500-5	501—126252	9.98 -7	1.00 -6	-7.52 -8	-2.16 -8	1155.1	1201.4
bqp500-6	501—126252	9.99 -7	9.99 -7	-7.37 -7	-8.01 -7	981.4	777.0
bqp500-7	501—126252	9.99 -7	9.99 -7	-1.07 -6	-1.56 -7	1116.2	914.2
bqp500-8	501—126252	9.99 -7	9.99 -7	-6.63 -7	-8.08 -7	1053.0	780.6
bqp500-9	501—126252	9.99 -7	9.99 -7	-1.05 -6	-1.54 -8	967.3	890.8
gka10b	126—8127	9.98 -7	9.97 -7	-4.19 -6	-8.90 -6	23.0	20.7
gka10d	101—5252	9.93 -7	9.93 -7	+6.59 -7	-7.02 -7	8.4	9.4
gka1d	101—5252	1.00 -6	9.99 -7	-2.33 -7	-1.60 -7	15.5	31.6
gka1e	201—20502	1.00 -6	1.00 -6	-2.34 -7	-3.41 -7	63.9	74.7
gka1f	501—126252	9.99 -7	1.00 -6	-8.67 -7	-5.93 -7	1012.0	871.1

gka2d	101—5252	9.98 -7	1.00 -6	-1.16 -7	-2.69 -7	8.1	16.3
gka2e	201—20502	9.98 -7	9.99 -7	-7.46 -7	-8.47 -7	49.1	57.7
gka2f	501—126252	9.99 -7	1.00 -6	-7.54 -7	-1.30 -6	1076.2	922.3
gka3d	101—5252	9.99 -7	1.00 -6	+1.66 -8	-4.88 -8	15.0	25.9
gka3e	201—20502	9.99 -7	1.00 -6	-1.55 -9	-1.40 -7	94.9	93.0
gka3f	501—126252	1.00 -6	1.00 -6	-9.60 -7	-6.97 -8	983.9	1023.6
gka4d	101—5252	9.93 -7	1.00 -6	+2.27 -7	-1.41 -7	7.0	17.7
gka4e	201—20502	1.00 -6	1.00 -6	-5.14 -7	-3.06 -7	67.7	83.7
gka4f	501—126252	1.00 -6	1.00 -6	-4.53 -7	-2.42 -7	1093.4	1165.5
gka5d	101—5252	9.93 -7	9.97 -7	-1.55 -7	-1.14 -7	7.1	10.2
gka5e	201—20502	9.99 -7	1.00 -6	-1.95 -8	-2.69 -8	80.6	99.1
gka5f	501—126252	1.00 -6	1.00 -6	-6.74 -7	-7.72 -7	973.9	746.1
gka6d	101—5252	9.97 -7	9.97 -7	-1.58 -8	-1.89 -8	10.0	15.5
gka7c	101—5252	9.96 -7	9.99 -7	-5.27 -7	-6.47 -7	15.2	44.6
gka7d	101—5252	9.94 -7	9.94 -7	-1.28 -6	-3.04 -7	7.0	9.2
gka8a	101—5252	9.99 -7	9.94 -7	+2.31 -7	+8.65 -7	87.0	40.8
gka8d	101—5252	9.97 -7	9.99 -7	-3.41 -7	-8.11 -8	13.6	27.9
gka9b	101—5252	9.74 -7	6.85 -7	+3.64 -7	-8.09 -6	4.0	7.1
gka9d	101—5252	1.00 -6	9.96 -7	+1.45 -6	-7.58 -8	7.2	8.2

Table 2.3.2: 2EBD-HPE results on BIQ problems

Instance	$n\|m$	$\langle c, x\rangle$	$\langle b, w\rangle$	**Iterations**	ϵ_P	ϵ_D	**Time**
be100.1	101—5252	-2.002134 +4	-2.002129 +4	1511	5.14 -7	9.69 -7	8.1
be100.10	101—5252	-1.640851 +4	-1.640853 +4	1232	1.20 -7	9.99 -7	7.1
be100.2	101—5252	-1.798870 +4	-1.798870 +4	1381	1.00 -6	4.79 -7	6.9
be100.3	101—5252	-1.823105 +4	-1.823106 +4	1619	9.98 -7	8.62 -7	9.6
be100.4	101—5252	-1.984180 +4	-1.984179 +4	1927	9.98 -7	4.85 -7	10.4
be100.5	101—5252	-1.688870 +4	-1.688871 +4	1286	1.00 -6	8.34 -7	7.2
be100.6	101—5252	-1.814822 +4	-1.814820 +4	1463	9.53 -7	1.00 -6	8.1
be100.7	101—5252	-1.970085 +4	-1.970080 +4	1379	2.91 -7	9.91 -7	7.5
be100.8	101—5252	-1.994642 +4	-1.994638 +4	1360	8.60 -7	9.95 -7	7.3
be100.9	101—5252	-1.426337 +4	-1.426336 +4	1191	7.74 -7	9.95 -7	7.4
be120.3.1	121—7502	-1.380356 +4	-1.380355 +4	1775	9.99 -7	3.21 -7	12.3
be120.3.10	121—7502	-1.293086 +4	-1.293085 +4	1394	9.96 -7	5.78 -7	9.8
be120.3.2	121—7502	-1.362663 +4	-1.362661 +4	1964	9.98 -7	6.80 -7	13.6
be120.3.3	121—7502	-1.298791 +4	-1.298789 +4	1551	8.05 -7	9.96 -7	11.2
be120.3.4	121—7502	-1.451125 +4	-1.451122 +4	1694	6.80 -7	9.95 -7	12.2
be120.3.5	121—7502	-1.199191 +4	-1.199191 +4	3558	1.00 -6	5.64 -7	27.1
be120.3.6	121—7502	-1.343206 +4	-1.343205 +4	2130	9.98 -7	7.95 -7	15.4
be120.3.7	121—7502	-1.456411 +4	-1.456411 +4	3809	1.00 -6	5.67 -7	27.6
be120.3.8	121—7502	-1.530302 +4	-1.530302 +4	2708	9.50 -7	1.00 -6	20.1
be120.3.9	121—7502	-1.124132 +4	-1.124131 +4	2616	4.26 -7	1.00 -6	18.5
be120.8.1	121—7502	-2.019393 +4	-2.019396 +4	1257	8.22 -7	9.95 -7	8.9
be120.8.10	121—7502	-2.002400 +4	-2.002400 +4	1551	2.75 -7	9.97 -7	10.7
be120.8.2	121—7502	-2.007413 +4	-2.007412 +4	2803	1.00 -6	4.13 -7	19.8
be120.8.3	121—7502	-2.050590 +4	-2.050590 +4	1524	9.96 -7	8.17 -7	11.5
be120.8.4	121—7502	-2.177980 +4	-2.177979 +4	2027	9.98 -7	2.79 -7	14.2
be120.8.5	121—7502	-2.131628 +4	-2.131628 +4	1743	9.96 -7	7.35 -7	12.1
be120.8.6	121—7502	-1.967696 +4	-1.967695 +4	1632	1.00 -6	3.41 -7	11.6
be120.8.7	121—7502	-2.373238 +4	-2.373238 +4	1677	6.16 -7	9.97 -7	11.7
be120.8.8	121—7502	-2.120478 +4	-2.120476 +4	1297	5.09 -7	9.98 -7	9.4
be120.8.9	121—7502	-1.928441 +4	-1.928441 +4	1274	4.72 -7	9.99 -7	10.0
be150.3.1	151—11627	-1.984918 +4	-1.984915 +4	2116	6.95 -7	9.97 -7	22.3
be150.3.10	151—11627	-1.923092 +4	-1.923091 +4	2768	9.99 -7	9.23 -7	30.0
be150.3.2	151—11627	-1.886485 +4	-1.886483 +4	2094	9.32 -7	9.98 -7	21.4
be150.3.3	151—11627	-1.804372 +4	-1.804370 +4	1757	8.70 -7	9.95 -7	18.8
be150.3.4	151—11627	-2.065267 +4	-2.065264 +4	2027	7.36 -7	9.97 -7	22.3

be150.3.5	151—11627	-1.776865 +4	-1.776865 +4	2589	1.00 -6	3.35 -8	27.6
be150.3.6	151—11627	-1.805069 +4	-1.805068 +4	1944	6.59 -7	9.98 -7	20.3
be150.3.7	151—11627	-1.910131 +4	-1.910129 +4	1947	9.43 -7	9.98 -7	21.2
be150.3.8	151—11627	-1.969806 +4	-1.969806 +4	2510	9.99 -7	2.79 -7	26.6
be150.3.9	151—11627	-1.410337 +4	-1.410335 +4	1190	4.40 -7	9.96 -7	12.3
be150.8.1	151—11627	-2.914369 +4	-2.914367 +4	1573	6.68 -7	9.95 -7	15.9
be150.8.10	151—11627	-3.004798 +4	-3.004796 +4	2043	9.99 -7	4.34 -7	22.2
be150.8.2	151—11627	-2.882110 +4	-2.882105 +4	1520	6.93 -7	9.98 -7	16.5
be150.8.3	151—11627	-3.106037 +4	-3.106036 +4	1821	9.98 -7	9.36 -7	19.9
be150.8.4	151—11627	-2.872930 +4	-2.872928 +4	2035	9.98 -7	3.56 -7	20.3
be150.8.5	151—11627	-2.948207 +4	-2.948204 +4	1991	9.91 -7	9.95 -7	22.3
be150.8.6	151—11627	-3.143723 +4	-3.143722 +4	2711	9.17 -7	9.61 -7	26.8
be150.8.7	151—11627	-3.325211 +4	-3.325206 +4	2470	8.81 -7	9.98 -7	28.3
be150.8.8	151—11627	-3.159999 +4	-3.159996 +4	2553	9.95 -7	6.77 -7	27.1
be150.8.9	151—11627	-2.711073 +4	-2.711071 +4	2931	9.99 -7	4.75 -7	30.6
be200.3.1	201—20502	-2.771609 +4	-2.771604 +4	2069	6.75 -7	9.97 -7	41.2
be200.3.10	201—20502	-2.576069 +4	-2.576068 +4	2345	9.99 -7	6.19 -7	45.6
be200.3.2	201—20502	-2.676079 +4	-2.676076 +4	2178	7.71 -7	9.97 -7	43.1
be200.3.3	201—20502	-2.947864 +4	-2.947862 +4	3554	1.00 -6	5.87 -7	65.8
be200.3.4	201—20502	-2.910621 +4	-2.910617 +4	2284	9.97 -7	9.98 -7	43.7
be200.3.5	201—20502	-2.807299 +4	-2.807296 +4	3289	6.87 -7	9.99 -7	60.5
be200.3.6	201—20502	-2.792835 +4	-2.792832 +4	1843	7.63 -7	1.00 -6	33.4
be200.3.7	201—20502	-3.162050 +4	-3.162048 +4	2638	5.92 -7	9.99 -7	53.3
be200.3.8	201—20502	-2.924429 +4	-2.924425 +4	2256	9.97 -7	8.32 -7	41.8
be200.3.9	201—20502	-2.643705 +4	-2.643700 +4	3923	9.06 -7	9.99 -7	74.1
be200.8.1	201—20502	-5.086949 +4	-5.086942 +4	2913	5.63 -7	9.98 -7	58.5
be200.8.10	201—20502	-4.574306 +4	-4.574301 +4	2131	9.99 -7	7.21 -7	43.0
be200.8.2	201—20502	-4.433604 +4	-4.433599 +4	1869	5.93 -7	9.97 -7	35.4
be200.8.3	201—20502	-4.625398 +4	-4.625392 +4	2656	8.07 -7	9.99 -7	48.9
be200.8.4	201—20502	-4.662125 +4	-4.662119 +4	2196	6.94 -7	9.99 -7	41.0
be200.8.5	201—20502	-4.427124 +4	-4.427122 +4	2257	9.99 -7	4.79 -7	41.6
be200.8.6	201—20502	-5.121888 +4	-5.121887 +4	3105	9.98 -7	3.75 -7	60.3
be200.8.7	201—20502	-4.935288 +4	-4.935266 +4	2099	6.28 -7	9.99 -7	39.9
be200.8.8	201—20502	-4.768917 +4	-4.768917 +4	2836	9.99 -7	2.61 -7	55.2
be200.8.9	201—20502	-4.549560 +4	-4.549558 +4	2820	9.96 -7	4.44 -7	57.3
be250.1	251—31877	-2.511946 +4	-2.511945 +4	4221	9.99 -7	4.95 -7	128.4
be250.10	251—31877	-2.435502 +4	-2.435501 +4	5469	1.00 -6	4.79 -7	168.8
be250.2	251—31877	-2.368149 +4	-2.368145 +4	3459	9.95 -7	9.99 -7	107.8
be250.3	251—31877	-2.400000 +4	-2.399999 +4	3443	1.00 -6	2.37 -7	104.8
be250.4	251—31877	-2.572032 +4	-2.572028 +4	6762	8.56 -7	9.96 -7	210.2

be250.5	251—31877	-2.237471 +4	-2.237468 +4	3996	1.00 -6	7.86 -7	121.7
be250.6	251—31877	-2.401885 +4	-2.401882 +4	3301	8.25 -7	9.98 -7	92.6
be250.7	251—31877	-2.511896 +4	-2.511895 +4	3844	1.00 -6	6.02 -7	117.7
be250.8	251—31877	-2.502040 +4	-2.502039 +4	3616	9.99 -7	3.32 -7	109.5
be250.9	251—31877	-2.139706 +4	-2.139704 +4	4891	9.90 -7	9.34 -7	146.1
bqp100-1	101—5252	-8.380388 +3	-8.380384 +3	1287	7.54 -7	9.99 -7	7.9
bqp100-10	101—5252	-1.298027 +4	-1.298027 +4	4546	1.00 -6	3.63 -7	24.6
bqp100-2	101—5252	-1.148926 +4	-1.148925 +4	3023	9.06 -7	9.98 -7	16.5
bqp100-3	101—5252	-1.315318 +4	-1.315318 +4	4229	1.00 -6	9.45 -7	23.1
bqp100-4	101—5252	-1.073189 +4	-1.073189 +4	2573	9.99 -7	9.13 -7	13.6
bqp100-5	101—5252	-9.487027 +3	-9.487026 +3	3577	1.00 -6	4.01 -7	19.1
bqp100-6	101—5252	-1.082474 +4	-1.082476 +4	1275	9.62 -7	9.96 -7	7.2
bqp100-7	101—5252	-1.068915 +4	-1.068913 +4	1819	7.07 -7	9.95 -7	10.1
bqp100-8	101—5252	-1.176999 +4	-1.176999 +4	3043	1.00 -6	6.38 -7	16.9
bqp100-9	101—5252	-1.173325 +4	-1.173325 +4	2747	9.99 -7	2.69 -7	15.9
bqp250-1	251—31877	-4.766311 +4	-4.766306 +4	3850	9.99 -7	4.12 -7	119.2
bqp250-10	251—31877	-4.301452 +4	-4.301442 +4	2741	6.28 -7	9.98 -7	84.0
bqp250-2	251—31877	-4.722238 +4	-4.722231 +4	3693	9.99 -7	9.96 -7	119.9
bqp250-3	251—31877	-5.107673 +4	-5.107655 +4	3636	4.83 -7	9.99 -7	116.4
bqp250-4	251—31877	-4.331256 +4	-4.331249 +4	2802	9.00 -7	9.99 -7	79.6
bqp250-5	251—31877	-5.000433 +4	-5.000424 +4	5559	6.93 -7	1.00 -6	166.5
bqp250-6	251—31877	-4.366886 +4	-4.366881 +4	4037	9.99 -7	6.28 -7	128.0
bqp250-7	251—31877	-4.892173 +4	-4.892164 +4	3543	1.00 -6	6.27 -7	107.4
bqp250-8	251—31877	-3.877955 +4	-3.877951 +4	2860	4.58 -7	9.98 -7	95.9
bqp250-9	251—31877	-5.149755 +4	-5.149751 +4	3988	9.99 -7	8.96 -7	118.5
bqp500-1	501—126252	-1.259642 +5	-1.259639 +5	6205	5.13 -7	9.99 -7	992.8
bqp500-10	501—126252	-1.385344 +5	-1.385340 +5	6299	5.69 -7	9.99 -7	1042.5
bqp500-2	501—126252	-1.360111 +5	-1.360108 +5	6821	5.55 -7	9.98 -7	1113.0
bqp500-3	501—126252	-1.384534 +5	-1.384530 +5	6359	4.56 -7	1.00 -6	1032.3
bqp500-4	501—126252	-1.393284 +5	-1.393280 +5	6435	4.21 -7	1.00 -6	970.1
bqp500-5	501—126252	-1.340921 +5	-1.340921 +5	7302	9.98 -7	1.22 -7	1155.1
bqp500-6	501—126252	-1.307644 +5	-1.307642 +5	6066	5.31 -7	9.99 -7	981.4
bqp500-7	501—126252	-1.314915 +5	-1.314912 +5	6503	5.11 -7	9.99 -7	1116.2
bqp500-8	501—126252	-1.334898 +5	-1.334897 +5	6567	5.18 -7	9.99 -7	1053.0
bqp500-9	501—126252	-1.302883 +5	-1.302880 +5	5911	7.50 -7	9.99 -7	967.3
gka10b	126—8127	-1.555721 +2	-1.555708 +2	3275	9.98 -7	1.53 -8	23.0
gka10d	101—5252	-2.010856 +4	-2.010859 +4	1423	8.89 -7	9.93 -7	8.4
gka1d	101—5252	-6.528429 +3	-6.528426 +3	2606	1.00 -6	7.59 -7	15.5
gka1e	201—20502	-1.706982 +4	-1.706981 +4	3380	1.00 -6	7.48 -7	63.9
gka1f	501—126252	-6.555908 +4	-6.555896 +4	6266	5.92 -7	9.99 -7	1012.0

gka2d	101—5252	-6.990710 +3	-6.990708 +3	1506	4.79 -7	9.98 -7	8.1
gka2e	201—20502	-2.491764 +4	-2.491760 +4	2471	8.77 -7	9.98 -7	49.1
gka2f	501—126252	-1.079318 +5	-1.079316 +5	6725	9.99 -7	6.80 -7	1076.2
gka3d	101—5252	-9.734332 +3	-9.734332 +3	2852	9.99 -7	5.44 -7	15.0
gka3e	201—20502	-2.689874 +4	-2.689874 +4	5080	9.99 -7	6.77 -7	94.9
gka3f	501—126252	-1.501510 +5	-1.501508 +5	5842	7.08 -7	1.00 -6	983.9
gka4d	101—5252	-1.127841 +4	-1.127842 +4	1341	9.48 -7	9.93 -7	7.0
gka4e	201—20502	-3.722515 +4	-3.722511 +4	3579	1.00 -6	8.10 -7	67.7
gka4f	501—126252	-1.870879 +5	-1.870877 +5	6421	1.00 -6	5.41 -7	1093.4
gka5d	101—5252	-1.239886 +4	-1.239886 +4	1334	9.93 -7	5.28 -7	7.1
gka5e	201—20502	-3.800231 +4	-3.800231 +4	4227	9.99 -7	1.63 -7	80.6
gka5f	501—126252	-2.069143 +5	-2.069140 +5	5573	5.08 -7	1.00 -6	973.9
gka6d	101—5252	-1.492936 +4	-1.492936 +4	1841	9.97 -7	1.77 -7	10.0
gka7c	101—5252	-7.316449 +3	-7.316441 +3	2924	9.96 -7	9.66 -7	15.2
gka7d	101—5252	-1.537582 +4	-1.537578 +4	1253	4.44 -7	9.94 -7	7.0
gka8a	101—5252	-1.119721 +4	-1.119722 +4	14987	5.59 -7	9.99 -7	87.0
gka8d	101—5252	-1.700536 +4	-1.700535 +4	2613	8.86 -7	9.97 -7	13.6
gka9b	101—5252	-1.369999 +2	-1.370000 +2	736	9.74 -7	5.34 -8	4.0
gka9d	101—5252	-1.653387 +4	-1.653391 +4	1270	8.73 -7	1.00 -6	7.2

2.3.4.2 Frequency assignment problems

This subsection gives more details of our implementation of 2EBD-HPE for solving SDP relaxations of FAPs and summarizes its computational performance against SDPAD on a collection of 7 such instances generated using a subroutine from SDPT3 described in [64]. Recall that the SDP relaxation of the FAP is described in (2.2.25) in Subsection 2.2.2.2.

There is more than one way of viewing (2.2.25) as a special case of formulation (2.3.31). For our current implementation, we have used the following two-easy-block structure formulation. Let $\mathcal{X} = \mathcal{W}_1 := \mathcal{S}^n$, $\mathcal{W}_2 = \mathbb{R}^{|E\backslash U|} \times \mathbb{R}^{|U|} \times \mathbb{R}^n$, $\mathcal{K}_1 = \mathcal{S}^n_+$ and $\mathcal{K}_2 = \mathbb{R}^{|E\backslash U|}_+ \times \mathbf{0}_{|U|} \times \mathbf{0}_n$, where $\mathbf{0}_n$ denotes an n dimensional vector of all zeros. With these definitions, we can easily see that (2.2.25) can be viewed as having the two-easy-block structure (2.3.31) if (2.2.25a) is chosen as $\mathcal{M}_1$, and (2.2.25b) and (2.2.25c) are chosen as $\mathcal{M}_2$. Note that, in view of the first inclusion in (2.3.37), the constraint $X \succeq 0$ is always satisfied by 2EBD-HPE, while SDPAD approaches it in the limit.

Table 2.3.3 compares the two methods on a collection of 7 SDP relaxations of FAPs.

Table 2.3.3: Comparison of the methods on FAPs

Problem		$\max\{\epsilon_P, \epsilon_D\}$		ϵ_G		Time	
Instance	$n_s \vert m$	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**
fap08	120—7260	9.30 -7	9.99 -7	-3.23 -6	-1.93 -6	5.7	6.4
fap09	174—15225	9.94 -7	9.98 -7	-3.07 -6	+5.10 -8	7.4	6.8
fap10	183—14479	1.96 -7	8.28 -7	-9.73 -6	-9.83 -6	76.2	174.9
fap11	252—24292	1.36 -7	1.12 -7	-9.99 -6	-1.00 -5	170.6	424.6
fap12	369—26462	1.57 -7	8.36 -8	-9.97 -6	-1.00 -5	556.4	1733.1
fap25	2118—322924	5.89 -7	1.42 -7	-9.68 -6	-1.00 -5	89519.7	258593.0
fap36	4110—1154467	6.62 -7	4.19 -7	-9.91 -6	-1.00 -5	293007.5	720433.8

Table 2.3.4: 2EBD-HPE results on FAPs

Instance	$n \vert m$	$\langle c, x \rangle$	$\langle b, w \rangle$	**Iterations**	ϵ_P	ϵ_D	**Time**
fap08	120—7260	+2.436266 +0	+2.436285 +0	956	9.30 -7	7.29 -7	5.7
fap09	174—15225	+1.079777 +1	+1.079784 +1	666	9.59 -7	9.94 -7	7.4
fap10	183—14479	+9.668432 -3	+9.678346 -3	4610	1.96 -7	1.63 -7	76.2
fap11	252—24292	+2.976395 -2	+2.977454 -2	5350	1.36 -7	9.90 -8	170.6
fap12	369—26462	+2.732333 -1	+2.732487 -1	8072	1.57 -7	7.39 -8	556.4
fap25	2118—322924	+1.287731 +1	+1.287757 +1	10270	5.89 -7	1.30 -7	89519.7
fap36	4110—1154467	+6.985624 +1	+6.985764 +1	5440	6.62 -7	4.10 -7	293007.5

For the purpose of this comparison, we have run 2EBD-HPE with $\sigma = 0.99$ and the values of γ, τ and $\bar{k}$ in the dynamic scaling rule (2.3.47) set to $\gamma = 1.5$, $\tau = 0.75$ and $\bar{k} = 5$. Table 2.3.4 gives more detailed computational results obtained by 2EBD-HPE (see the second paragraph preceding Subsection 2.3.4.1 for an explanation on this table). Figure 2.3.5 plots the performance profiles of both methods on this collection of 7 SDP relaxations of FAPs.

Note that 2EBD-HPE solves 6 (out of a total of 7) problems faster than SDPAD. Moreover, 2EBD-HPE solves about 5 of them, including the two largest ones (i.e., `fap25` and `fap36`), at least 2 times faster than SDPAD.

2.3.4.3 SDPs arising from relaxation of maximum stable set problems

This subsection gives more details of our implementation of 2EBD-HPE for solving SDPs corresponding to θ-functions and θ_+-functions of graph stable set problems and summarizes its computational performance against SDPAD on a collection of 58 θ-function SDP instances and the corresponding collection of 58 θ_+-function SDP instances. Recall that the SDPs for θ-functions and θ_+-functions of graph stable set problems are described in (2.2.28) in Subsection 2.2.2.5.

There is more than one way of viewing the $\theta(G)$ and $\theta_+(G)$ problems as special cases of formulation (2.3.31). For our current implementation, we have used the following two-easy-block structure formulations. For the case of the $\theta(G)$ (resp. $\theta_+(G)$) problem, we

let $\mathcal{X} = \mathcal{S}^n$, $\mathcal{W}_1 := \mathcal{S}^n \times \mathbb{R}$, $\mathcal{W}_2 = \mathbb{R} \times \mathbb{R}^{|E|}$, $\mathcal{K}_1 = \mathcal{S}^n_+ \times \mathbf{0}_1$ and $\mathcal{K}_2 = \mathbf{0}_1 \times \mathbf{0}_{|E|}$ (resp. $\mathcal{K}_2 = \mathbf{0}_1 \times \mathbf{0}_{|E|} \times \mathbb{R}^{n(n+1)/2}_+$). With these definitions, we can easily see that the $\theta(G)$ and $\theta_+(G)$ problems can be viewed as having the two-easy-block structure (2.3.31) if $\mathcal{M}_1$ (resp. $\mathcal{M}_2$) is chosen to be the set of $X \in \mathcal{S}^n$ satisfying (2.2.28a) and (2.2.28b) (resp. (2.2.28b) and (2.2.28c)). Note that (2.2.28b) is used to define both $\mathcal{M}_1$ and $\mathcal{M}_2$. Note that, in view of the first inclusion in (2.3.37), the constraints $X \succeq 0$ and $I \bullet X = 1$ are always satisfied by 2EBD-HPE, while SDPAD approaches them in the limit.

Tables 2.3.5 and 2.3.7 compare the two methods on a collection of 58 $\theta(G)$ instances and the corresponding collection of 58 $\theta_+(G)$ instances, respectively. For the purpose of this comparison, we have run 2EBD-HPE with $\sigma = 0.9$ and the values of γ, τ and $\bar{k}$ in the dynamic scaling rule (2.3.47) set to $\gamma = 1.5$, $\tau = 0.75$ and $\bar{k} = 5$. For the $\theta(G)$ problems, we have used the safeguard that the dynamic scaling scheme is not performed at those iterations k for which the first inequality in (2.3.44) is satisfied with $\bar{\epsilon} = 10^{-5}$. Tables 2.3.6 and 2.3.8 give more detailed computational results obtained by 2EBD-HPE (see the second paragraph preceding Subsection 2.3.4.1 for an explanation on this table). Figures 2.3.7 and 2.3.8 plot the performance profiles of both methods for solving $\theta(G)$ and $\theta_+(G)$, respectively, on this collection of 58 $\theta(G)$ instances and the corresponding collection of 58 $\theta_+(G)$ instances.

Note that 2EBD-HPE solves 36 (out of a total of 58) $\theta(G)$ and 49 (out of a total of 58) $\theta_+(G)$ problems faster than SDPAD. Moreover, 2EBD-HPE solves about 7 $\theta(G)$ and 12 $\theta_+(G)$ problems at least 4 times faster than SDPAD. Note also that 2EBD-HPE fails to solve 5 $\theta(G)$ and 1 $\theta_+(G)$ instances while SDPAD fails to solve 5 $\theta(G)$ and 6 $\theta_+(G)$ instances.

2.3.5 Numerical results: part II

In this section, we briefly compare 2EBD-HPE with the SDPNAL method presented in [68] and a BD method presented in [35], namely DSA-BD. We use for this comparison the same four problem classes described in Section 2.3.4.

In contrast to 2EBD-HPE, the methods DSA-BD and SDPNAL always require as input

Table 2.3.5: Comparison of the methods on $\theta(G)$

Problem		$\max\{\epsilon_P, \epsilon_D\}$		ϵ_G		Time	
Instance	$n_s \vert m$	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**
1dc.1024	1024—24064	1.00 -6	1.00 -6	-9.34 -6	-6.57 -6	7437.6	10208.3
1dc.128	128—1472	1.00 -6	1.82 -6*	-6.21 -6	+6.13 -6	86.9	464.9*
1dc.2048	2048—58368	7.45 -7	7.63 -7	-1.00 -5	-1.00 -5	107634.6	134043.4
1dc.256	256—3840	9.53 -7	9.98 -7	-3.14 -6	+5.16 -6	60.2	170.3
1dc.512	512—9728	1.00 -6	1.00 -6	-8.58 -6	-8.78 -6	1192.0	1341.1
1et.1024	1024—9601	9.41 -7	9.51 -7	-8.64 -6	-1.00 -5	5685.5	9460.2
1et.128	128—673	9.92 -7	8.54 -7	-3.92 -7	+1.64 -6	3.7	3.3
1et.2048	2048—22529	3.62 -3*	1.00 -6	+3.18 -2*	-9.37 -6	159336.8*	110786.1
1et.256	256—1665	9.99 -7	9.99 -7	-2.53 -6	-1.25 -6	60.8	136.3
1et.512	512—4033	9.95 -7	9.61 -7	-5.04 -6	-2.80 -6	504.2	1254.1
1tc.1024	1024—7937	9.93 -7	1.00 -6	-6.35 -6	-9.09 -6	14874.0	19483.4
1tc.128	128—513	8.96 -7	9.79 -7	-4.07 -7	-7.04 -8	2.5	6.3
1tc.2048	2048—18945	2.44 -3*	9.94 -7	+2.22 -2*	-1.00 -5	156775.0*	156132.9
1tc.256	256—1313	1.00 -6	1.00 -6	-1.49 -6	-2.36 -6	171.3	284.4
1tc.512	512—3265	9.98 -7	1.00 -6	-4.95 -6	-6.28 -6	3233.4	3565.0
1zc.1024	1024—16641	8.74 -7	9.17 -7	-8.10 -6	-1.58 -6	1520.1	1194.7
1zc.128	128—1121	7.88 -7	9.43 -7	+9.82 -6	+1.81 -6	2.0	1.9
1zc.256	256—2817	5.25 -7	8.23 -7	+7.53 -6	-2.35 -6	7.3	10.2
1zc.512	512—6913	7.48 -7	9.32 -7	+7.20 -6	+1.55 -6	83.2	120.9
2dc.1024	1024—169163	3.04 -7	2.45 -6*	+1.48 -5*	+6.20 -5*	23295.4*	157809.8*
2dc.512	512—54896	9.94 -7	9.64 -7	-1.58 -6	+1.25 -6	2713.1	17410.2
G43	1000—9991	9.96 -7	9.76 -7	-1.61 -6	+1.81 -6	969.0	894.2
G44	1000—9991	9.93 -7	9.31 -7	-1.19 -6	+8.75 -7	1024.8	949.8
G45	1000—9991	9.94 -7	9.61 -7	-1.18 -6	-1.77 -6	1044.2	1102.5
G46	1000—9991	9.96 -7	9.88 -7	-1.12 -6	+1.55 -6	997.4	1014.9
G47	1000—9991	9.90 -7	9.48 -7	-9.87 -7	-8.88 -7	1114.4	894.1
G51	1000—5910	1.00 -6	1.11 -6*	-5.19 -7	+4.26 -7	6164.0	20923.4*
G52	1000—5917	2.30 -6*	4.14 -6*	+2.20 -5*	+2.24 -5*	21841.7*	17074.8*
G53	1000—5915	2.40 -6*	4.08 -6*	+1.79 -5*	+3.24 -5*	21511.2*	19416.6*
G54	1000—5917	9.99 -7	1.00 -6	-4.81 -7	-1.46 -6	4554.3	9943.4
brock200-1	200—5067	9.69 -7	9.28 -7	-7.52 -7	+2.19 -7	5.7	5.4
brock200-4	200—6812	9.84 -7	9.86 -7	-8.09 -7	-3.25 -8	5.2	4.3
brock400-1	400—20078	9.56 -7	9.40 -7	-8.14 -7	-1.06 -6	27.7	32.3
c-fat200-1	200—18367	9.74 -7	9.89 -7	-2.20 -6	+3.47 -6	3.4	28.5
hamming-10-2	1024—23041	9.76 -7	9.63 -7	-8.99 -6	-2.18 -6	1189.1	1123.1
hamming-7-5-6	128—1793	9.85 -7	8.77 -7	+5.53 -6	+1.43 -6	1.3	4.0
hamming-8-3-4	256—16129	4.95 -7	5.27 -7	-8.92 -6	-9.57 -7	3.5	13.7
hamming-8-4	256—11777	8.46 -7	7.71 -7	+9.97 -6	-2.22 -6	6.4	7.6
hamming-9-5-6	512—53761	9.37 -7	9.76 -7	-9.11 -6	+1.81 -6	25.7	914.7
hamming-9-8	512—2305	7.41 -7	9.70 -7	+9.97 -6	-5.63 -7	160.4	477.8
keller4	171—5101	9.82 -7	9.88 -7	-8.69 -7	+1.65 -6	3.7	5.2
p-hat300-1	300—33918	9.98 -7	9.98 -7	-4.03 -6	-2.98 -6	34.4	236.0
san200-0.7-1	200—5971	9.93 -7	9.92 -7	-2.52 -6	-2.37 -6	3.0	135.9
sanr200-0.7	200—6033	9.72 -7	9.33 -7	-6.04 -7	+2.00 -7	5.4	4.9
theta10	500—12470	9.86 -7	9.87 -7	-8.00 -7	+1.36 -6	64.6	113.2
theta102	500—37467	9.69 -7	9.10 -7	-1.04 -6	-1.70 -6	55.5	61.3
theta103	500—62516	9.75 -7	9.88 -7	-1.35 -6	-2.01 -7	50.4	102.9
theta104	500—87245	9.85 -7	9.54 -7	-3.61 -6	-8.34 -7	49.4	239.1
theta12	600—17979	9.79 -7	8.52 -7	-7.26 -7	-1.01 -6	114.4	99.3
theta123	600—90020	9.86 -7	9.19 -7	-2.53 -6	-2.23 -7	98.6	201.4
theta32	150—2286	9.89 -7	9.94 -7	-4.64 -7	-5.04 -7	3.9	3.7
theta4	200—1949	9.84 -7	9.58 -7	-6.42 -7	+5.93 -7	8.6	6.2
theta42	200—5986	9.70 -7	9.93 -7	-6.14 -7	-2.70 -7	6.0	4.8
theta6	300—4375	9.79 -7	9.49 -7	-5.25 -7	+1.12 -6	18.6	15.3
theta62	300—13390	9.83 -7	9.65 -7	-1.31 -6	-1.09 -6	13.9	12.2
theta8	400—7905	9.70 -7	8.25 -7	-6.52 -7	-9.81 -7	36.4	31.5
theta82	400—23872	9.85 -7	9.56 -7	-9.01 -7	-1.96 -6	28.7	30.1
theta83	400—39862	9.67 -7	9.18 -7	-1.21 -6	-2.65 -7	29.7	51.7

Table 2.3.6: 2EBD-HPE results on $\theta(G)$

Instance	$n\|m$	$\langle c,x\rangle$	$\langle b,w\rangle$	**Iterations**	ϵ_P	ϵ_D	**Time**
1dc.1024	1024—24064	-9.598719 +1	-9.598538 +1	6657	1.00 -6	6.18 -7	7437.6
1dc.128	128—1472	-1.684216 +1	-1.684194 +1	10375	9.79 -7	1.00 -6	86.9
1dc.2048	2048—58368	-1.747333 +2	-1.747298 +2	14660	7.45 -7	5.55 -7	107634.6
1dc.256	256—3840	-3.000019 +1	-3.000000 +1	2001	9.53 -7	9.01 -7	60.2
1dc.512	512—9728	-5.303177 +1	-5.303085 +1	8332	1.00 -6	8.53 -7	1192.0
1et.1024	1024—9601	-1.842298 +2	-1.842266 +2	6477	8.48 -7	9.41 -7	5685.5
1et.128	128—673	-2.923093 +1	-2.923091 +1	544	9.73 -7	9.92 -7	3.7
1et.2048	2048—22529	-3.233531 +2	-3.445938 +2	20000	6.45 -4	3.62 -3	159336.8
1et.256	256—1665	-5.511451 +1	-5.511422 +1	2274	9.28 -7	9.99 -7	60.8
1et.512	512—4033	-1.044250 +2	-1.044240 +2	3303	9.95 -7	8.55 -7	504.2
1tc.1024	1024—7937	-2.063072 +2	-2.063046 +2	13504	7.76 -7	9.93 -7	14874.0
1tc.128	128—513	-3.800005 +1	-3.800002 +1	414	8.96 -7	7.48 -7	2.5
1tc.2048	2048—18945	-3.598373 +2	-3.762285 +2	20000	5.82 -4	2.44 -3	156775.0
1tc.256	256—1313	-6.340007 +1	-6.339988 +1	5784	7.68 -7	1.00 -6	171.3
1tc.512	512—3265	-1.134015 +2	-1.134003 +2	19047	8.16 -7	9.98 -7	3233.4
1zc.1024	1024—16641	-1.286689 +2	-1.286668 +2	1891	5.61 -7	8.74 -7	1520.1
1zc.128	128—1121	-2.066625 +1	-2.066666 +1	344	7.88 -7	6.18 -7	2.0
1zc.256	256—2817	-3.799942 +1	-3.800000 +1	335	5.25 -7	4.10 -7	7.3
1zc.512	512—6913	-6.874906 +1	-6.875005 +1	592	6.39 -7	7.48 -7	83.2
2dc.1024	1024—169163	-1.863852 +1	-1.863795 +1	20000	3.04 -7	2.51 -7	23295.4
2dc.512	512—54896	-1.176785 +1	-1.176781 +1	15790	1.83 -7	9.94 -7	2713.1
G43	1000—9991	-2.806255 +2	-2.806246 +2	877	3.98 -7	9.96 -7	969.0
G44	1000—9991	-2.805839 +2	-2.805832 +2	930	4.01 -7	9.93 -7	1024.8
G45	1000—9991	-2.801858 +2	-2.801852 +2	922	3.98 -7	9.94 -7	1044.2
G46	1000—9991	-2.798376 +2	-2.798370 +2	893	4.05 -7	9.96 -7	997.4
G47	1000—9991	-2.818945 +2	-2.818940 +2	933	4.22 -7	9.90 -7	1114.4
G51	1000—5910	-3.490004 +2	-3.490001 +2	6110	1.00 -6	8.58 -7	6164.0
G52	1000—5917	-3.484100 +2	-3.483946 +2	20000	2.30 -6	1.87 -6	21841.7
G53	1000—5915	-3.483644 +2	-3.483519 +2	20000	1.86 -6	2.40 -6	21511.2
G54	1000—5917	-3.410003 +2	-3.410000 +2	4092	6.51 -7	9.99 -7	4554.3
brock200-1	200—5067	-2.745668 +1	-2.745664 +1	288	3.94 -7	9.69 -7	5.7
brock200-4	200—6812	-2.129351 +1	-2.129348 +1	264	3.86 -7	9.84 -7	5.2
brock400-1	400—20078	-3.970197 +1	-3.970190 +1	300	3.20 -7	9.56 -7	27.7
c-fat200-1	200—18367	-1.200003 +1	-1.199998 +1	286	4.73 -7	9.74 -7	3.4
hamming-10-2	1024—23041	-1.024019 +2	-1.024001 +2	1197	1.57 -7	9.76 -7	1189.1
hamming-7-5-6	128—1793	-4.266616 +1	-4.266664 +1	262	9.85 -7	7.85 -7	1.3
hamming-8-3-4	256—16129	-2.560046 +1	-2.559999 +1	173	4.95 -7	2.20 -7	3.5
hamming-8-4	256—11777	-1.599969 +1	-1.600002 +1	284	3.28 -7	8.46 -7	6.4
hamming-9-5-6	512—53761	-8.533495 +1	-8.533339 +1	203	8.59 -7	9.37 -7	25.7
hamming-9-8	512—2305	-2.239954 +2	-2.239999 +2	1280	7.41 -7	3.72 -7	160.4
keller4	171—5101	-1.401226 +1	-1.401223 +1	330	4.35 -7	9.82 -7	3.7
p-hat300-1	300—33918	-1.006806 +1	-1.006797 +1	700	9.98 -7	6.63 -7	34.4
san200-0.7-1	200—5971	-3.000000 +1	-2.999985 +1	169	4.93 -8	9.93 -7	3.0
sanr200-0.7	200—6033	-2.383619 +1	-2.383616 +1	269	3.85 -7	9.72 -7	5.4
theta10	500—12470	-8.380611 +1	-8.380597 +1	401	3.48 -7	9.86 -7	64.6
theta102	500—37467	-3.839063 +1	-3.839055 +1	297	3.23 -7	9.69 -7	55.5
theta103	500—62516	-2.252863 +1	-2.252857 +1	287	3.10 -7	9.75 -7	50.4
theta104	500—87245	-1.333624 +1	-1.333614 +1	302	5.41 -7	9.85 -7	49.4
theta12	600—17979	-9.280182 +1	-9.280169 +1	410	3.07 -7	9.79 -7	114.4
theta123	600—90020	-2.466878 +1	-2.466865 +1	299	5.00 -7	9.86 -7	98.6
theta32	150—2286	-2.757160 +1	-2.757157 +1	352	6.23 -7	9.89 -7	3.9
theta4	200—1949	-5.032129 +1	-5.032122 +1	469	4.71 -7	9.84 -7	8.6
theta42	200—5986	-2.393174 +1	-2.393171 +1	292	4.90 -7	9.70 -7	6.0
theta6	300—4375	-6.347716 +1	-6.347709 +1	401	3.82 -7	9.79 -7	18.6
theta62	300—13390	-2.964133 +1	-2.964125 +1	284	5.32 -7	9.83 -7	13.9
theta8	400—7905	-7.395367 +1	-7.395357 +1	389	3.42 -7	9.70 -7	36.4
theta82	400—23872	-3.436696 +1	-3.436689 +1	287	3.37 -7	9.85 -7	28.7
theta83	400—39862	-2.030194 +1	-2.030189 +1	278	3.20 -7	9.67 -7	29.7

Table 2.3.7: Comparison of the methods on $\theta_+(G)$

Problem		$\max\{\epsilon_P, \epsilon_D\}$		ϵ_G		Time	
Instance	$n_s \vert m$	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**	**2EBD-HPE**	**SDPAD**
1dc.1024	1024—24064	1.00 -6	1.00 -6	-3.32 -6	-1.41 -7	3819.0	11958.7
1dc.128	128—1472	9.89 -7	9.99 -7	-2.89 -6	-2.93 -7	8.1	46.3
1dc.2048	2048—58368	1.00 -6	3.35 -6*	-4.91 -6	+7.44 -7	59560.7	235634.3*
1dc.256	256—3840	8.19 -7	9.94 -7	-3.31 -6	-5.14 -6	11.2	234.7
1dc.512	512—9728	9.99 -7	9.99 -7	-1.65 -6	-1.37 -7	400.7	1616.0
1et.1024	1024—9601	1.00 -6	9.99 -7	-3.74 -6	-3.81 -7	2429.3	7186.4
1et.128	128—673	9.87 -7	9.85 -7	-7.40 -8	+1.97 -8	3.6	4.7
1et.2048	2048—22529	1.00 -6	1.00 -6	-6.22 -6	-2.56 -7	30816.3	149449.2
1et.256	256—1665	3.69 -6*	1.00 -6	+8.63 -7	-1.06 -7	689.6*	106.9
1et.512	512—4033	9.99 -7	9.98 -7	-3.17 -6	-2.78 -7	176.6	593.3
1tc.1024	1024—7937	9.95 -7	2.57 -6*	-3.59 -6	+4.25 -7	6879.1	29837.5*
1tc.128	128—513	8.90 -7	9.94 -7	-3.04 -6	+3.48 -8	1.6	6.9
1tc.2048	2048—18945	9.97 -7	3.57 -6*	-5.20 -6	+6.42 -7	55981.8	168205.8*
1tc.256	256—1313	1.00 -6	1.00 -6	-1.02 -6	-1.36 -7	84.7	168.4
1tc.512	512—3265	1.00 -6	1.00 -6	-2.14 -6	-9.91 -8	560.0	2448.3
1zc.1024	1024—16641	9.83 -7	8.94 -7	+8.40 -6	+5.38 -7	1834.9	1220.2
1zc.128	128—1121	9.01 -7	9.81 -7	+7.75 -6	-1.36 -6	1.6	2.1
1zc.256	256—2817	8.97 -7	9.96 -7	-6.56 -6	-1.20 -6	6.8	10.8
1zc.512	512—6913	8.22 -7	9.89 -7	+1.14 -6	+1.11 -10	140.7	338.0
2dc.1024	1024—169163	9.96 -7	1.00 -6	-4.66 -6	-2.73 -7	2394.4	85897.6
2dc.512	512—54896	9.66 -7	1.00 -6	-9.99 -6	-2.77 -7	326.9	4058.2
G43	1000—9991	9.98 -7	9.37 -7	-1.71 -6	-1.67 -6	772.3	1002.7
G44	1000—9991	9.98 -7	9.91 -7	-1.45 -6	-1.62 -6	804.6	1091.7
G45	1000—9991	9.98 -7	9.10 -7	-1.40 -6	+6.19 -7	801.9	1025.9
G46	1000—9991	9.98 -7	9.29 -7	-1.23 -6	+5.70 -7	948.0	1125.1
G47	1000—9991	9.97 -7	9.76 -7	-1.11 -6	+4.96 -7	840.7	1039.9
G51	1000—5910	9.99 -7	5.63 -6*	-2.91 -7	+5.19 -8	6521.5	20902.5*
G52	1000—5917	9.94 -7	4.35 -5*	-4.91 -7	+1.16 -6	9781.8	22086.4*
G53	1000—5915	1.00 -6	5.15 -5*	-3.31 -6	+8.46 -6	21959.6	22189.0*
G54	1000—5917	9.92 -7	9.99 -7	-9.40 -7	-1.11 -8	3274.6	10437.7
brock200-1	200—5067	9.98 -7	9.83 -7	-9.48 -7	-3.33 -8	6.1	7.1
brock200-4	200—6812	9.98 -7	9.82 -7	-1.23 -6	-5.42 -8	5.7	5.7
brock400-1	400—20078	9.68 -7	9.97 -7	-1.18 -6	+2.37 -7	31.7	27.8
c-fat200-1	200—18367	8.71 -7	9.45 -7	-2.43 -6	-9.66 -7	3.5	21.9
hamming-10-2	1024—23041	7.82 -7	8.55 -7	+7.96 -6	-3.49 -6	1276.1	947.9
hamming-7-5-6	128—1793	6.20 -7	9.17 -7	+9.41 -6	-4.53 -7	4.0	7.3
hamming-8-3-4	256—16129	9.68 -7	9.99 -7	+9.60 -6	+1.10 -6	3.9	10.4
hamming-8-4	256—11777	9.31 -7	9.66 -7	-8.06 -6	-2.13 -6	5.4	5.9
hamming-9-5-6	512—53761	9.20 -7	9.99 -7	+8.37 -6	+1.88 -6	74.3	247.4
hamming-9-8	512—2305	8.60 -7	9.16 -7	-7.73 -6	+5.12 -7	115.4	383.5
keller4	171—5101	9.86 -7	9.95 -7	-1.19 -6	+1.31 -7	6.7	20.9
p-hat300-1	300—33918	9.94 -7	9.99 -7	-1.03 -6	-9.93 -8	34.0	697.3
san200-0.7-1	200—5971	9.73 -7	9.61 -7	-1.17 -6	+4.02 -6	2.7	101.7
sanr200-0.7	200—6033	9.63 -7	9.99 -7	-4.25 -7	-5.65 -8	6.1	7.2
theta10	500—12470	9.80 -7	9.18 -7	-9.45 -7	-1.39 -6	68.6	65.5
theta102	500—37467	9.96 -7	9.74 -7	-1.49 -6	-2.65 -7	52.0	70.3
theta103	500—62516	9.84 -7	9.80 -7	-1.80 -6	-1.33 -7	52.8	109.9
theta104	500—87245	9.87 -7	9.97 -7	-2.18 -6	-2.15 -7	49.3	262.8
theta12	600—17979	9.65 -7	9.90 -7	-9.33 -7	-1.04 -6	128.3	104.1
theta123	600—90020	9.96 -7	9.82 -7	-1.73 -6	-1.17 -7	93.2	226.3
theta32	150—2286	9.93 -7	9.97 -7	-3.62 -7	-2.34 -8	4.2	5.9
theta4	200—1949	9.76 -7	9.91 -7	-5.28 -7	-1.28 -7	8.7	10.3
theta42	200—5986	9.87 -7	9.92 -7	-4.35 -7	-5.31 -8	6.3	8.4
theta6	300—4375	1.00 -6	9.93 -7	-3.83 -7	-5.71 -7	21.6	21.0
theta62	300—13390	9.96 -7	9.84 -7	-9.46 -7	-1.09 -7	14.8	15.9
theta8	400—7905	9.88 -7	8.97 -7	-9.37 -7	-1.67 -6	39.2	32.9
theta82	400—23872	9.98 -7	9.93 -7	-1.33 -6	+5.39 -8	29.7	31.7
theta83	400—39862	9.75 -7	9.92 -7	-1.59 -6	-1.41 -7	27.8	46.1

Table 2.3.8: 2EBD-HPE results on $\theta_+(G)$

Instance	$n\|m$	$\langle c,x\rangle$	$\langle b,w\rangle$	**Iterations**	ϵ_P	ϵ_D	**Time**
1dc.1024	1024—24064	-9.555185 +1	-9.555121 +1	3058	5.86 -7	1.00 -6	3819.0
1dc.128	128—1472	-1.667839 +1	-1.667829 +1	945	5.93 -7	9.89 -7	8.1
1dc.2048	2048—58368	-1.742593 +2	-1.742575 +2	6634	4.87 -7	1.00 -6	59560.7
1dc.256	256—3840	-3.000003 +1	-2.999983 +1	374	5.46 -7	8.19 -7	11.2
1dc.512	512—9728	-5.269531 +1	-5.269514 +1	2350	4.35 -7	9.99 -7	400.7
1et.1024	1024—9601	-1.820729 +2	-1.820715 +2	2194	4.07 -7	1.00 -6	2429.3
1et.128	128—673	-2.923091 +1	-2.923090 +1	517	9.87 -7	7.92 -7	3.6
1et.2048	2048—22529	-3.381694 +2	-3.381652 +2	4125	5.50 -7	1.00 -6	30816.3
1et.256	256—1665	-5.446508 +1	-5.446499 +1	20000	3.69 -6	3.22 -6	689.6
1et.512	512—4033	-1.035499 +2	-1.035492 +2	1221	4.47 -7	9.99 -7	176.6
1tc.1024	1024—7937	-2.042055 +2	-2.042041 +2	6510	6.84 -7	9.95 -7	6879.1
1tc.128	128—513	-3.800018 +1	-3.799995 +1	221	6.03 -7	8.90 -7	1.6
1tc.2048	2048—18945	-3.704924 +2	-3.704886 +2	6472	6.29 -7	9.97 -7	55981.8
1tc.256	256—1313	-6.324048 +1	-6.324035 +1	2627	3.91 -7	1.00 -6	84.7
1tc.512	512—3265	-1.125343 +2	-1.125338 +2	3093	5.11 -7	1.00 -6	560.0
1zc.1024	1024—16641	-1.279977 +2	-1.279999 +2	1852	2.58 -7	9.83 -7	1834.9
1zc.128	128—1121	-2.066632 +1	-2.066665 +1	250	8.24 -7	9.01 -7	1.6
1zc.256	256—2817	-3.733380 +1	-3.733330 +1	270	5.73 -7	8.97 -7	6.8
1zc.512	512—6913	-6.799987 +1	-6.800002 +1	938	4.02 -7	8.22 -7	140.7
2dc.1024	1024—169163	-1.771014 +1	-1.770997 +1	2348	3.01 -7	9.96 -7	2394.4
2dc.512	512—54896	-1.138372 +1	-1.138349 +1	2092	4.96 -7	9.66 -7	326.9
G43	1000—9991	-2.797370 +2	-2.797360 +2	777	5.68 -7	9.98 -7	772.3
G44	1000—9991	-2.797469 +2	-2.797461 +2	814	5.75 -7	9.98 -7	804.6
G45	1000—9991	-2.793184 +2	-2.793176 +2	819	5.55 -7	9.98 -7	801.9
G46	1000—9991	-2.790332 +2	-2.790325 +2	792	5.69 -7	9.98 -7	948.0
G47	1000—9991	-2.808923 +2	-2.808917 +2	827	5.70 -7	9.97 -7	840.7
G51	1000—5910	-3.490001 +2	-3.489999 +2	5834	5.55 -7	9.99 -7	6521.5
G52	1000—5917	-3.483865 +2	-3.483862 +2	7511	5.21 -7	9.94 -7	9781.8
G53	1000—5915	-3.482137 +2	-3.482114 +2	18374	6.22 -7	1.00 -6	21959.6
G54	1000—5917	-3.410004 +2	-3.409997 +2	3260	6.05 -7	9.92 -7	3274.6
brock200-1	200—5067	-2.719677 +1	-2.719672 +1	299	6.09 -7	9.98 -7	6.1
brock200-4	200—6812	-2.112113 +1	-2.112108 +1	271	5.96 -7	9.98 -7	5.7
brock400-1	400—20078	-3.933102 +1	-3.933092 +1	307	5.08 -7	9.68 -7	31.7
c-fat200-1	200—18367	-1.200004 +1	-1.199998 +1	263	7.46 -7	8.71 -7	3.5
hamming-10-2	1024—23041	-8.533190 +1	-8.533327 +1	1119	1.87 -7	7.82 -7	1276.1
hamming-7-5-6	128—1793	-3.599932 +1	-3.600001 +1	666	6.20 -7	2.04 -7	4.0
hamming-8-3-4	256—16129	-2.559948 +1	-2.559998 +1	180	9.68 -7	6.31 -7	3.9
hamming-8-4	256—11777	-1.600024 +1	-1.599998 +1	225	2.52 -7	9.31 -7	5.4
hamming-9-5-6	512—53761	-5.866561 +1	-5.866660 +1	593	6.08 -7	9.20 -7	74.3
hamming-9-8	512—2305	-2.240033 +2	-2.239998 +2	968	5.36 -7	8.60 -7	115.4
keller4	171—5101	-1.346593 +1	-1.346590 +1	519	6.15 -7	9.86 -7	6.7
p-hat300-1	300—33918	-1.002023 +1	-1.002021 +1	697	4.52 -7	9.94 -7	34.0
san200-0.7-1	200—5971	-3.000000 +1	-2.999993 +1	152	9.17 -8	9.73 -7	2.7
sanr200-0.7	200—6033	-2.363331 +1	-2.363329 +1	293	3.25 -7	9.63 -7	6.1
theta10	500—12470	-8.314916 +1	-8.314900 +1	414	5.04 -7	9.80 -7	68.6
theta102	500—37467	-3.806637 +1	-3.806625 +1	308	5.18 -7	9.96 -7	52.0
theta103	500—62516	-2.237750 +1	-2.237742 +1	288	4.84 -7	9.84 -7	52.8
theta104	500—87245	-1.328267 +1	-1.328261 +1	287	4.48 -7	9.87 -7	49.3
theta12	600—17979	-9.209105 +1	-9.209088 +1	421	4.43 -7	9.65 -7	128.3
theta123	600—90020	-2.449524 +1	-2.449515 +1	293	4.32 -7	9.96 -7	93.2
theta32	150—2286	-2.729164 +1	-2.729162 +1	346	4.23 -7	9.93 -7	4.2
theta4	200—1949	-4.986907 +1	-4.986902 +1	458	3.91 -7	9.76 -7	8.7
theta42	200—5986	-2.373823 +1	-2.373821 +1	291	3.99 -7	9.87 -7	6.3
theta6	300—4375	-6.296189 +1	-6.296185 +1	431	3.31 -7	1.00 -6	21.6
theta62	300—13390	-2.937800 +1	-2.937794 +1	286	4.48 -7	9.96 -7	14.8
theta8	400—7905	-7.340799 +1	-7.340785 +1	400	5.49 -7	9.88 -7	39.2
theta82	400—23872	-3.406444 +1	-3.406435 +1	300	5.32 -7	9.98 -7	29.7
theta83	400—39862	-2.016717 +1	-2.016711 +1	279	4.85 -7	9.75 -7	27.8

a conic optimization problem given in standard form, i.e., as in (1.1.1). Hence, it is necessary for the latter two codes it is necessary (except for the θ-function SDP problems) to add additional variables and/or constraints to the original conic optimization problem (2.3.31) in order to obtain a standard form formulation. Thus, the number of variables and/or constraints handled by the latter two codes are usually larger than the number of variables and/or constraints handled by 2EBD-HPE. As the computational results of this section show, this has a negative effect on the performance of DSA-BD and SDPNAL compared to 2EBD-HPE. In fact, the main goal of the benchmark presented in this section is to show that taking advantage of any special structure of the original conic SDP formulation of the problem results in much more efficient codes both in terms of computation time and RAM.

For the 2EBD-HPE, DSA-BD and SDPNAL methods, the computational results for the SDP relaxations of BIQs and FAPs were obtained on a server with 2 Xeon X5460 processors at 3.16GHz and 32GB RAM, and the ones corresponding to the SDPs for θ-functions and θ_+-functions of graph stable set problems were obtained on a single core of a server with 2 Xeon X5520 processors at 2.27GHz and 48GB RAM.

For this benchmark, we have adopted the same stopping criterion as the one used in [30, 52], [35] and [68] to compare the three methods. More specifically, all methods are stopped whenever

$$\max\{\epsilon_{P,k}, \epsilon_{D,k}\} \leq \bar{\epsilon},$$

with $\bar{\epsilon} = 10^{-6}$. Even though we could have incorporated $\epsilon_{G,k}$ in the termination criterion for this benchmark, we decided to leave it out as has been done in the benchmarks of [30, 52], [35] and [68].

For the sake of shortness, we only report the performance profiles and exclude the detailed tables as the ones reported in Section 2.3.4. Figures 2.3.9, 2.3.10 and 2.3.11 plot the performance profiles of 2EBD-HPE, DSA-BD and SDPNAL for the SDP relaxations of BIQ problems, the SDP relaxations of FAPs, and the SDPs for θ-functions and θ_+-functions of graph stable set problems, respectively. Note that based on these performance profiles, 2EBD-HPE outperforms DSA-BD and SDPNAL in every problem class.

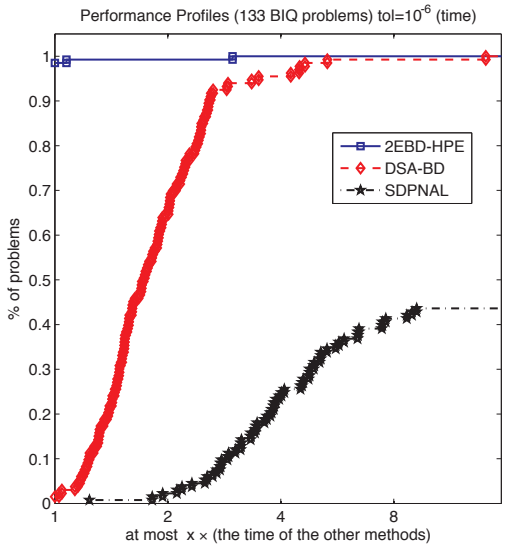

Figure 2.3.9: Performance profiles of 2EBD-HPE, DSA-BD and SDPNAL for solving 133 SDP relaxations of BIQ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

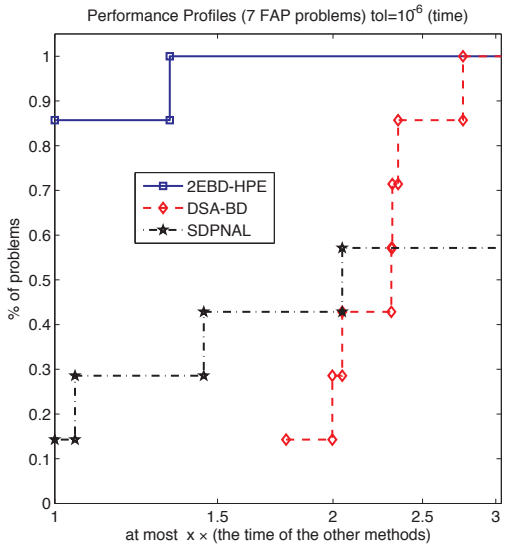

Figure 2.3.10: Performance profiles of 2EBD-HPE, the BD method in [35] and SDPNAL for solving 7 SDP relaxations of FAPs with accuracy $\bar{\epsilon} = 10^{-6}$.

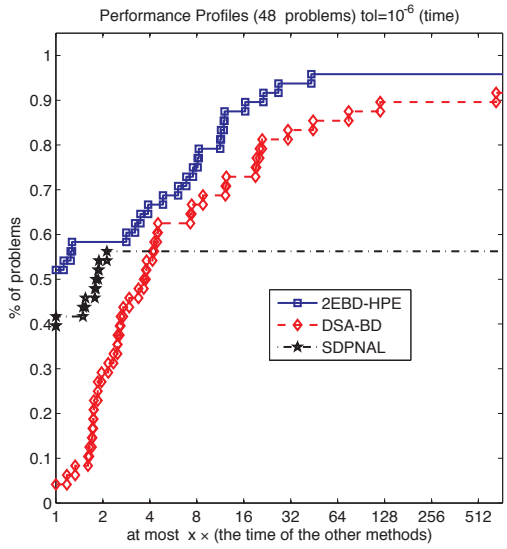

Figure 2.3.11: Performance profiles of 2EBD-HPE, the BD method in [35] and SDPNAL for solving 48 $\theta(G)$ and $\theta_+(G)$ problems with accuracy $\bar{\epsilon} = 10^{-6}$.

2.4 Concluding remarks

Note that when applying the A-BD-HPE framework to (2.3.9), it is necessary to first specify the first and second blocks, namely $0 \in F_1(x,y)+C(x)$ and $0 \in F_2(x,y)+D(x)$, respectively. We have seen that Algorithm 2.2 corresponds to applying the A-BD-HPE framework to (2.3.9) by choosing the first and second blocks to be the first and second inclusions in (2.3.9), respectively. Clearly, a variant of Algorithm 2.2 can be obtained by changing the choice of the first and second blocks to be the second and first inclusions in (2.3.9), respectively. The resulting method can be easily shown to possess similar convergence properties as those of Algorithm 2.2. We observe that $\tilde{\lambda}$ for this variant should be chosen as

$$\tilde{\lambda} := \min\left\{\frac{\sigma_1^2}{\theta L_f}, \frac{(\sigma^2-\sigma_1^2)^{1/2}}{\sqrt{\theta}}\right\}.$$

The approach in Section 2.3.1 can be easily extended to the convex problem

$$\begin{array}{ll} \min & f(x)+\sum_{i=0}^m h_i(x) \\ \text{s.t.} & x \in \mathcal{X}, \end{array} \tag{2.4.1}$$

which is equivalent to solving the inclusion problem

$$0 \in \nabla f(x) + \partial h_0(x) + \sum_{i=1}^m y_i,$$

$$0 \in \theta_i[-x + \partial h_i^*(y_i)], \quad i=1,\ldots,m,$$

where $\theta_i > 0$, $i = 1,\ldots,m$, are scaling factors. Even though, this inclusion system has $m+1$ blocks of inclusions, it can be viewed as having two blocks for the purpose of applying the A-BD-HPE framework to it. Indeed, the first block would be the first inclusion and the second block would consist of the other m inclusions. Note that once $\tilde{x}^k$ is obtained from the proximal equation associated with the first block, it can be updated in the proximal equations corresponding to the other inclusions, and the $\tilde{y}_i^k$ can all be computed simultaneously. Convergence results similar to the ones obtained in Section 2.3.1 can be derived for (2.4.1) using the general convergence theory for BD type methods developed in [39].

Finally, our implementation of DSA-BD and 2EBD-HPE can be found at `http://www.isye.gatech.edu/~cod3/COrtiz/software/`. Although in this chapter we have only reported computational results for problems of the form (2.3.31), it should be mentioned that

the current version of 2EBD-HPE is capable of solving problems of the general form (2.3.1).

Chapter III

INEXACT BLOCK-DECOMPOSITION METHODS FOR CONIC PROGRAMMING

In this chapter, we present an inexact first-order BD method for solving standard form conic programming which avoids computations of exact projections onto the manifold defined by the affine constraints and, as a result, is able to handle extra large SDP instances. Section 3.1 presents an inexact first-order instance of the A-BD-HPE framework, and corresponding iteration-complexity results, for solving the conic programming problem (1.1.1) which avoids the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1.1). Section 3.2 describes a practical variant of the inexact BD method of Section 3.1 which incorporates a new dynamic scaling scheme and the use of the CG method to inexactly solve an augmented linear system. This latter scheme generalizes the dynamic scaling ideas used in Chapter 2 by using two scaling factors (instead of one) that change dynamically to balance three blocks of inclusions that comprise the optimality conditions for (1.1.1). Section 2.3.4 presents numerical results comparing SDPLR with the BD variant studied in this chapter on a collection of extra large-scale conic SDP instances. Finally, Section 3.4 presents some final remarks.

3.1 An inexact scaled BD algorithm for conic programming

In this section, we introduce an inexact instance of the A-BD-HPE framework applied to (1.1.1) which avoids the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1.1), and defines $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$ as scaled inner products constructed by means of the original inner products $\langle\cdot,\cdot\rangle$ of $\mathcal{X}$ and $\mathcal{Y}$. (Recall from Section 1.4 that the original inner products in $\mathcal{X}$ and $\mathcal{Y}$ used in (1.1.1) are both being denoted by $\langle\cdot,\cdot\rangle$.)

We consider problem (1.1.1) satisfying assumptions D.1 and D.2 defined in Subsection 2.2.1.

We now make a few observations about the assumptions D.1 and D.2. First, Assumption

D.2 is equivalent to the existence of $y^* \in \mathcal{Y}$ and $z^* \in \mathcal{X}$ such that the triple (z^*, y^*, x^*) satisfies

$$\begin{aligned} 0 &\in x^* + N_{K^*}(z^*) = x^* + \partial\delta_{K^*}(z^*), && \text{(3.1.1a)} \\ 0 &= \mathcal{A}x^* - b, && \text{(3.1.1b)} \\ 0 &= c - \mathcal{A}^* y^* - z^*. && \text{(3.1.1c)} \end{aligned}$$

Second, it is well-known that a triple (z^*, y^*, x^*) satisfies (3.1.1) if and only if x^* is an optimal solution of (1.1.1), the pair (z^*, y^*) is an optimal solution of (1.1.2) and duality gap between (1.1.1) and (1.1.2) is zero, i.e., $\langle c, x^* \rangle = \langle b, y^* \rangle$.

Our main goal in this section is to present an instance of the A-BD-HPE framework which approximately solves (3.1.1) *and does not require computation of exact projections onto the manifold* $\mathcal{M}$ (see O.3 in Section 1.1). Instead of dealing directly with (3.1.1), it is more efficient from a computational point of view to consider its equivalent scaled reformulation

$$\begin{aligned} 0 &\in \theta\left(x^* + N_{K^*}(z^*)\right) = \theta x^* + N_{K^*}(z^*), && \text{(3.1.2a)} \\ 0 &= \mathcal{A}x^* - b, && \text{(3.1.2b)} \\ 0 &= \xi(c - \mathcal{A}^* y^* - z^*), && \text{(3.1.2c)} \end{aligned}$$

where θ and ξ are positive scalars.

We will now show that (3.1.2) can be viewed as a special instance of (2.1.2) which satisfies assumptions A.1–A.4. Indeed, let

$$\mathcal{Z} = \mathcal{X}, \qquad \mathcal{W} = \mathcal{Y} \times \mathcal{X}, \tag{3.1.3}$$

and define the inner products as

$$\langle \cdot, \cdot \rangle_{\mathcal{Z}} := \theta^{-1}\langle \cdot, \cdot \rangle, \quad \langle (y, x), (y', x') \rangle_{\mathcal{W}} := \langle y, y' \rangle + \xi^{-1}\langle x, x' \rangle \quad \forall x, x' \in \mathcal{X},\ \forall y, y' \in \mathcal{Y}, \tag{3.1.4}$$

and the operators F, C and D as

$$F(z,y,x)=\begin{bmatrix}\theta x\\ \mathcal{A}x-b\\ \xi(c-\mathcal{A}^*y-z)\end{bmatrix},\qquad C(z)=\partial\delta_{K^*}(z),\qquad D(y,x)=(0,0),\tag{3.1.5}$$

for every $(z,y,x)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{X}$.

The following proposition can be easily shown.

Proposition 3.1. *The inclusion problem* (3.1.2) *is equivalent to the inclusion problem* (2.1.2) *where the spaces $\mathcal{Z}$ and $\mathcal{W}$, the inner products $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, and the operators F, C and D are defined as in* (3.1.3), (3.1.4) *and* (3.1.5). *Moreover, assumptions A.1–A.4 of Section 2.1.2 hold with $L=\sqrt{\theta\xi}$, and, as a consequence,* (3.1.2) *is maximal monotone with respect to $\langle\cdot,\cdot\rangle_{\mathcal{Z}\times\mathcal{W}}$* (see (2.1.1)).

In view of Proposition 3.1, from now on we consider the context in which (3.1.2) is viewed as a special case of (2.1.2) with $\mathcal{Z}$, $\mathcal{W}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$, $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, F, C and D given by (3.1.3), (3.1.4) and (3.1.5). In what follows, we present an instance of the A-BD-HPE framework in this particular context which solves the first block (2.1.4) exactly (i.e., with $\sigma_z=\tilde{\sigma}_z=\varepsilon_k^z=0$), and solves the second block (2.1.6) inexactly (i.e., with $\sigma_w>0$ and $\varepsilon_k^w=0$) with the help of a linear solver.

More specifically, let $\{(z^k,w^k)\}$, $\{(\tilde{z}^k,\tilde{w}^k)\}$ and $\{(\tilde{c}^k,\tilde{d}^k)\}$ denote the sequences as in the A-BD-HPE framework specialized to the above context with $\sigma_z=\tilde{\sigma}_z=\varepsilon_k^z=\varepsilon_k^w=0$. For every $k\in\mathbb{N}$, in view of (3.1.3), we have that $z^k,\tilde{z}^k\in\mathcal{X}$, and w^k and $\tilde{w}^k$ can be written as $w^k=:(y^k,x^k)$ and $\tilde{w}^k=:(\tilde{y}^k,\tilde{x}^k)$, respectively, where $y^k,\tilde{y}^k\in\mathcal{Y}$ and $x^k,\tilde{x}^k\in\mathcal{X}$. Also, the fact that $\sigma_z=\tilde{\sigma}_z=\varepsilon_k^z=0$ implies that (2.1.4) in step 1 is equivalent to

$$\tilde{\lambda}[\theta x^{k-1}+\tilde{c}^k]+\tilde{z}^k-z^{k-1}=0,\qquad \tilde{c}^k\in\partial\delta_{K^*}(\tilde{z}^k)=N_{K^*}(\tilde{z}^k),\tag{3.1.6}$$

and hence, that $\tilde{z}^k$ satisfies the optimality conditions of the problem

$$\min_{z\in K^*}\frac{1}{2}\left\|z-\left[z^{k-1}-\tilde{\lambda}\theta x^{k-1}\right]\right\|^2.$$

Therefore, we conclude that $\tilde{z}^k$ and $\tilde{c}^k$ are uniquely determined by

$$\tilde{z}^k = \Pi_{K^*}(z^{k-1} - \tilde{\lambda}\theta x^{k-1}), \qquad \tilde{c}^k = -\theta x^{k-1} + \frac{(z^{k-1} - \tilde{z}^k)}{\tilde{\lambda}}. \tag{3.1.7}$$

Moreover, we can easily see that (2.1.6) in step 2 is equivalent to setting

$$(\tilde{y}^k, \tilde{x}^k) = (y^{k-1}, x^{k-1}) + \Delta^k, \qquad \tilde{d}^k = (0, 0), \tag{3.1.8}$$

where the displacement $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfies

$$\|\mathcal{Q}\Delta^k - q^k\|_{\mathcal{W}} \leq \sigma_w \|\Delta^k\|_{\mathcal{W}}, \tag{3.1.9}$$

and, the linear mapping $\mathcal{Q} : \mathcal{Y} \times \mathcal{X} \to \mathcal{Y} \times \mathcal{X}$ and the vector $q^k \in \mathcal{Y} \times \mathcal{X}$ are defined as

$$\mathcal{Q} := \begin{bmatrix} \mathcal{I} & \tilde{\lambda}\mathcal{A} \\ -\tilde{\lambda}\xi\mathcal{A}^* & \mathcal{I} \end{bmatrix}, \qquad q^k := \tilde{\lambda} \begin{bmatrix} b - \mathcal{A}x^{k-1} \\ \xi\left(\mathcal{A}^* y^{k-1} + \tilde{z}^k - c\right) \end{bmatrix}.$$

Observe that finding Δ^k satisfying (3.1.9) with $\sigma_w = 0$ is equivalent to solving augmented primal-dual linear system $\mathcal{Q}\Delta = q^k$ *exactly*, which can be easily seen to be at least as difficult as projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1.1). Instead, the approach outlined above assumes $\sigma_w > 0$ and *inexactly* solves this augmented linear system by allowing a relative error as in (3.1.9). Clearly a Δ^k satisfying (3.1.9) can be found with the help of a suitable iterative linear solver.

We now state an instance of the Special-BD framework based on the ideas outlined above.

Algorithm 3.1 Inexact scaled BD method for (1.1.1)

0) Let $(z^0, y^0, x^0) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{X}$, $\theta, \xi > 0$, $\sigma_w \in [0, 1)$ and $\sigma \in (\sigma_w, 1]$ be given, and set

$$\tilde{\lambda} := \sqrt{\frac{\sigma^2 - \sigma_w^2}{\theta\xi}}, \tag{3.1.10}$$

and $k = 1$;

1) set $\tilde{z}^k = \Pi_{K^*}(z^{k-1} - \tilde{\lambda}\theta x^{k-1})$;

2) use a linear solver to find $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfying (3.1.9), and set

$$(\tilde{y}^k, \tilde{x}^k) = (y^{k-1}, x^{k-1}) + \Delta^k;$$

3) choose λ_k to be the largest $\lambda > 0$ such that

$$\left\|\lambda(v_z^k, v_y^k, v_x^k) + (\tilde{z}^k, \tilde{y}^k, \tilde{x}^k) - (z^{k-1}, y^{k-1}, x^{k-1})\right\|_{\mathcal{Z}\times\mathcal{W}} \leq \sigma \left\|(\tilde{z}^k, \tilde{y}^k, \tilde{x}^k) - (z^{k-1}, y^{k-1}, x^{k-1})\right\|_{\mathcal{Z}\times\mathcal{W}},$$

where

$$v_z^k = \theta(\tilde{x}^k - x^{k-1}) + \frac{(z^{k-1} - \tilde{z}^k)}{\tilde{\lambda}}, \qquad v_y^k = \mathcal{A}\tilde{x}^k - b, \qquad v_x^k = \xi(c - \mathcal{A}^*\tilde{y}^k - \tilde{z}^k); \tag{3.1.11}$$

4) set $(z^k, y^k, x^k) = (z^{k-1}, y^{k-1}, x^{k-1}) - \lambda_k(v_z^k, v_y^k, v_x^k)$ and $k \leftarrow k + 1$, and go to step 1.

We now make a few observations about Algorithm 3.1 and its relationship with the A-BD-HPE framework in the context of (3.1.3), (3.1.4) and (3.1.5). First, it is easy to check that $\tilde{\lambda}$ as in (3.1.10) satisfies (2.1.3) with $\sigma_z = \tilde{\sigma}_z = 0$ and $L = \sqrt{\theta\xi}$ as equality. Second, the discussion preceding Algorithm 3.1 shows that steps 1 and 2 of Algorithm 3.1 are equivalent to the same ones of the A-BD-HPE framework with $\sigma_z = 0$. Third, noting that (3.1.5), (3.1.7) and (3.1.8) imply that

$$(v_z^k, v_y^k, v_x^k) = F(\tilde{z}^k, \tilde{y}^k, \tilde{x}^k) + (h_1^k, h_2^k), \tag{3.1.12}$$

it follows that steps 3 and 4 of Algorithm 3.1 are equivalent to the same ones of the A-BD-HPE framework. Fourth, in view of the previous two observations, Algorithm 3.1

is a special instance of the A-BD-HPE framework for solving (2.1.2) where $\mathcal{Z}$, $\mathcal{W}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$, $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, F, h_1 and h_2 are given by (3.1.3), (3.1.4) and (3.1.5). Fifth, λ_k in step 3 of Algorithm 3.1 can be obtained by solving an easy quadratic equation. Finally, in Subsection 3.2.1 we discuss how the CG method is used in our implementation to obtain a vector Δ^k satisfying (3.1.9).

We now specialize Theorem 2.7 to the context of Algorithm 3.1. Even though Algorithm 3.1 is an instance of the A-BD-HPE framework applied to the scaled inclusion (3.1.2), the convergence result below is stated with respect to the unscaled inclusion (3.1.1).

Theorem 3.2. *Consider the sequences $\{(z^k, y^k, x^k)\}$, $\{(\tilde{z}^k, \tilde{y}^k, \tilde{x}^k)\}$ and $\{(v_z^k, v_y^k, v_x^k)\}$ generated by Algorithm 3.1 under the assumption that $\sigma < 1$ and, for every $k \in \mathbb{N}$, define*

$$r_z^k := \theta^{-1} v_z^k, \qquad r_y^k := v_x^k, \qquad r_x^k := \xi^{-1} v_y^k. \tag{3.1.13}$$

Let $P^* \in \mathcal{X}$ and $D^* \subseteq \mathcal{X} \times \mathcal{Y}$ denote the set of optimal solutions of (1.1.1) and (1.1.2), respectively, and define

$$\begin{aligned} d_{0,P} &:= \min\{\|x - x^0\| : x \in P^*\}, \\ d_{0,D} &:= \min\{\|(z,y) - (z^0,y^0)\| : (z,y) \in D^*\}. \end{aligned}$$

Then, for every $k \in \mathbb{N}$,

$$r_z^k \in \tilde{x}^k + N_{K^*}(\tilde{z}^k), \tag{3.1.14a}$$

$$r_y^k = \mathcal{A}\tilde{x}^k - b. \tag{3.1.14b}$$

$$r_x^k = c - \mathcal{A}^*\tilde{y}^k - \tilde{z}^k, \tag{3.1.14c}$$

and there exists $i \leq k$ such that

$$\sqrt{\theta\|r_z^i\|^2 + \|r_y^i\|^2 + \xi\|r_x^i\|^2} \leq \sqrt{\frac{\xi\theta}{k\,(\sigma^2 - \sigma_w^2)}}\sqrt{\left(\frac{1+\sigma}{1-\sigma}\right)\left(\max\{1, \theta^{-1}\} d_{0,D}^2 + \xi^{-1} d_{0,P}^2\right)}. \tag{3.1.15}$$

Proof. Consider the sequences $\{\tilde{c}^k\}$ and $\{\tilde{d}^k\}$ defined in (3.1.7) and (3.1.8). Identities (3.1.14b) and (3.1.14c) follow immediately from the two last identities in both (3.1.11)

and (3.1.13). Also, it follows from the inclusion in (3.1.6) and the definitions of $\tilde{c}^k$, v_z^k and r_z^k in (3.1.7), (3.1.11) and (3.1.13), respectively, that

$$r_z^k = \theta^{-1} v_z^k = \tilde{x}^k + \theta^{-1}\tilde{c}^k \in \tilde{x}^k + N_{K^*}(\tilde{z}^k),$$

and hence, that (3.1.14a) holds. Let d_0 denote the distance of $((z^0, y^0), x^0)$ to $D^* \times P^*$ with respect to the scaled norm $\|\cdot\|_{\mathcal{Z}\times\mathcal{W}}$, and observe that (3.1.4) and, the definitions of $d_{0,P}$ and $d_{0,D}$, imply that

$$d_0 \leq \sqrt{\max\{1, \theta^{-1}\} d_{0,D}^2 + \xi^{-1} d_{0,P}^2}. \tag{3.1.16}$$

Moreover, (3.1.12) together with Theorem 2.7 imply the existence of $i \leq k$ such that

$$\|(v_z^i, v_y^i, v_x^i)\|_{\mathcal{Z}\times\mathcal{W}} \leq \sqrt{\frac{1+\sigma}{1-\sigma}} \frac{d_0}{\tilde{\lambda}\sqrt{k}}. \tag{3.1.17}$$

Now, combining (3.1.17) with the definitions of $\|\cdot\|_{\mathcal{Z}\times\mathcal{W}}$, $\langle\cdot,\cdot\rangle_{\mathcal{Z}}$, $\langle\cdot,\cdot\rangle_{\mathcal{W}}$, r_z^k, r_y^k and r_x^k in (2.1.1), (3.1.4) and (3.1.13) , we have

$$\sqrt{\theta\|r_z^i\|^2 + \|r_y^i\|^2 + \xi\|r_x^i\|^2} \leq \sqrt{\frac{1+\sigma}{1-\sigma}} \frac{d_0}{\tilde{\lambda}\sqrt{k}},$$

which, together with (3.1.16) and the definition of $\tilde{\lambda}$ in (3.1.10), imply (3.1.15). □

We now make some observations about Algorithm 3.1 and Theorem 3.2. First, the pointwise iteration-complexity bound in Theorem 3.2 is $\mathcal{O}(1/\sqrt{k})$. It is possible to derive an $\mathcal{O}(1/k)$ iteration-complexity ergodic bound for Algorithm 3.1 as an immediate consequence of Theorem 3.3 of [39] and Theorem 2.4 of [35]. Second, the bound in (3.1.15) of Theorem 3.2 sheds light on how the scaling parameters θ and ξ might affect the sizes of the residuals r_z^i, r_y^i and r_x^i. Roughly speaking, viewing all quantities in (3.1.15), with the exception of θ, as constants, we see that

$$\left\|r_z^i\right\| = \mathcal{O}\left(\max\left\{1, \theta^{-1/2}\right\}\right), \qquad \max\left\{\left\|r_y^i\right\|, \left\|r_x^i\right\|\right\} = \mathcal{O}\left(\max\left\{1, \theta^{1/2}\right\}\right).$$

Hence, the ratio $\mathcal{R}_\theta := \max\left\{\left\|r_y^i\right\|, \left\|r_x^i\right\|\right\} / \left\|r_z^i\right\|$ can grow significantly as $\theta \to \infty$, while it can become very small as $\theta \to 0$. This suggests that $\mathcal{R}_\theta$ increases (resp., decreases) as ξ increases (resp., decreases). Similarly, viewing all quantities in the bound (3.1.15), with the exception of ξ, as constants, we see that

$$\left\|r_x^i\right\| = \mathcal{O}\left(\max\left\{1, \xi^{-1/2}\right\}\right), \qquad \left\|r_y^i\right\| = \mathcal{O}\left(\max\left\{1, \xi^{1/2}\right\}\right).$$

Hence, the ratio $\mathcal{R}_\xi := \left\|r_y^i\right\| / \left\|r_x^i\right\|$ can grow significantly as $\xi \to \infty$, while it can become very small as $\xi \to 0$. This suggests that $\mathcal{R}_\xi$ increases (resp., decreases) as ξ increases (resp., decreases). In fact, we have observed in our computational experiments that the ratios $\mathcal{R}_\theta$ and $\mathcal{R}_\xi$ behave just as described. Finally, observe that while the dual iterate $\tilde{z}^k$ is in K^*, the primal iterate $\tilde{x}^k$ is not necessarily in K. However, the following result shows that it is possible to construct a primal iterate $\tilde{u}^k$ which lies in K, is orthogonal to $\tilde{z}^k$ (i.e., $\tilde{u}^k$ and $\tilde{z}^k$ are complementary) and $\liminf_{k\to\infty} \|\mathcal{A}\tilde{u}^k - b\| = 0$.

Corollary 3.3. *Consider the same assumptions as in Theorem 3.2 and, for every $k \in \mathbb{N}$, define*

$$\tilde{u}^k := \tilde{x}^k - r_z^k, \qquad r_u^k := \mathcal{A}\tilde{u}^k - b. \tag{3.1.18}$$

Then, for every $k \in \mathbb{N}$, the following statements hold:

a) $\tilde{u}^k \in K$, $\tilde{z}^k \in K^*$ and $\langle \tilde{u}^k, \tilde{z}^k \rangle = 0$;

b) there exists $i \leq k$ such that

$$\|r_u^i\| \leq \sqrt{\frac{\xi \max\{\theta, \|\mathcal{A}\|^2\}}{k\,(\sigma^2 - \sigma_w^2)}} \sqrt{\left(\frac{1+\sigma}{1-\sigma}\right)\left(\max\{1, \theta^{-1}\} d_{0,D}^2 + \xi^{-1} d_{0,P}^2\right)}. \tag{3.1.19}$$

Proof. To show a), observe that inclusion (3.1.14a) and the definition of $\tilde{u}^k$ in (3.1.18) imply $\tilde{u}^k \in -N_{K^*}(\tilde{z}^k)$, and hence a) follows. Statement b) follows from (3.1.15) and the fact that (3.1.14b) and (3.1.18) imply

$$\|r_u^i\| = \|r_y^i - \mathcal{A}r_z^i\| \leq \|r_y^i\| + \|\mathcal{A}r_z^i\| \leq \max\{1, \theta^{-1/2}\|\mathcal{A}\|\} \cdot \sqrt{\theta\|r_z^i\|^2 + \|r_y^i\|^2}.$$

□

3.2 A practical dynamically scaled inexact BD method

This section is divided into two subsections. The first one introduces a practical procedure that uses an iterative linear solver for computing Δ^k as in step 2 of Algorithm 3.1. The second one describes the stopping criterion used for Algorithm 3.1 and presents two important refinements of Algorithm 3.1 based on the ideas introduced in [35] and [33] that allow the scaling parameters ξ and θ to change dynamically.

3.2.1 Solving step 2 of Algorithm 3.1 for large and/or dense problems

In this subsection, we present a procedure that uses an iterative linear solver for computing Δ^k as in step 2 of Algorithm 3.1. More specifically, we show how the CG method applied to a linear system with an $m \times m$ positive definite symmetric coefficient matrix yields a displacement $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfying (3.1.9), where $m = \dim \mathcal{Y}$.

In order to describe the procedure for computing $\Delta^k \in \mathcal{Y} \times \mathcal{X}$ satisfying (3.1.9), define $\tilde{\mathcal{Q}} : \mathcal{Y} \to \mathcal{Y}$ and $\tilde{q}^k \in \mathcal{Y}$ as

$$\tilde{\mathcal{Q}} := \mathcal{I} + \tilde{\lambda}^2 \xi \mathcal{A}\mathcal{A}^*, \tag{3.2.1}$$

$$\tilde{q}^k := \tilde{\lambda}\left(b - \mathcal{A}x^{k-1}\right) - \tilde{\lambda}^2 \xi \mathcal{A}\left(\mathcal{A}^* y^{k-1} + \tilde{z}^k - c\right) \tag{3.2.2}$$

and observe that $\tilde{\mathcal{Q}}$ is a self-adjoint positive definite linear operator. The CG method can then be applied to to the linear system $\tilde{\mathcal{Q}}\Delta_y = \tilde{q}^k$ to obtain a solution $\Delta_y^k \in \mathcal{Y}$ satisfying

$$\|\tilde{\mathcal{Q}}\Delta_y^k - \tilde{q}^k\| \leq \sigma_w \|\Delta_y^k\|. \tag{3.2.3}$$

(In our implementation, we choose $\Delta = 0$ as the initial point for the CG method.) Setting

$$\Delta^k = (\Delta_y^k, \Delta_x^k)$$

where

$$\Delta_x^k = \tilde{\lambda}\xi\left(\mathcal{A}^*\left(y^{k-1} + \Delta_y^k\right) + \tilde{z}^k - c\right),$$

we can easily check that Δ^k satisfy (3.1.9).

We now make some observations about the above procedure for computing the displacement Δ^k. First, the arithmetic complexity of an iteration of the CG method corresponds to that of performing single evaluations of the operators $\mathcal{A}$ and $\mathcal{A}^*$. For example, if $\mathcal{X}$ and $\mathcal{Y}$ are given by (1.1.4) and $\mathcal{A}$ is identified with and stored as a sparse matrix, then the arithmetic complexity of an iteration of the CG method is bounded by $\mathcal{O}(\text{nnz}(\mathcal{A}))$. As another example, if $\mathcal{A}$ is the product of two matrices $\mathcal{A}_1$ and $\mathcal{A}_2$ where $\text{nnz}(\mathcal{A}_1)$ and $\text{nnz}(\mathcal{A}_2)$ are significantly smaller than $\text{nnz}(\mathcal{A})$, the bound on the arithmetic complexity of an iteration of the CG method can be improved to $\mathcal{O}(\text{nnz}(\mathcal{A}_1) + \text{nnz}(\mathcal{A}_2))$. Second, in view of (3.2.1) and (3.1.10), we have $\tilde{\mathcal{Q}} = \mathcal{I} + (\sigma^2 - \sigma_w^2)\theta^{-1}\mathcal{A}\mathcal{A}^*$. Hence, assuming that σ_w^{-1} is $\mathcal{O}(1)$, and

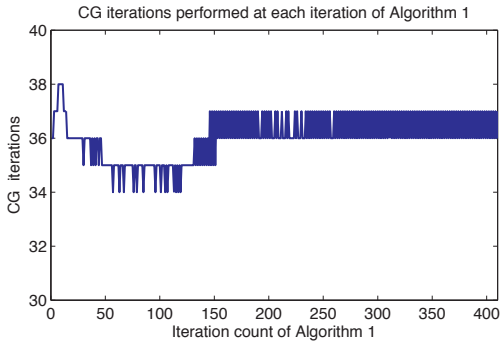

Figure 3.2.1: This example (random SDP instance) illustrates how the number of iterations of the CG method does not change significantly from one iteration of Algorithm 3.1 to the next when scaling parameter θ remains constant.

defining $\lambda_{\max}^{\mathcal{A}} := \lambda_{\max}(\mathcal{A}\mathcal{A}^*)$ and $\lambda_{\min}^{\mathcal{A}} := \lambda_{\min}(\mathcal{A}\mathcal{A}^*)$, it follows from Proposition A.4 with $\delta = \sigma_w$ that the CG method will compute Δ_y^k satisfying (3.2.3) in

$$\mathcal{O}\left(\sqrt{\kappa(\tilde{\mathcal{Q}})}\log\left[\sqrt{\kappa(\tilde{\mathcal{Q}})}\left(1+\|\tilde{\mathcal{Q}}\|\right)\right]\right) = \mathcal{O}\left(\sqrt{\frac{\theta+\lambda_{\max}^{\mathcal{A}}}{\theta+\lambda_{\min}^{\mathcal{A}}}}\log\left[\sqrt{\frac{\theta+\lambda_{\max}^{\mathcal{A}}}{\theta+\lambda_{\min}^{\mathcal{A}}}}\left(1+\frac{\lambda_{\max}^{\mathcal{A}}}{\theta}\right)\right]\right) \quad (3.2.4)$$

iterations, where the equality follows from the fact that $\|\tilde{Q}\| = \mathcal{O}(1+\lambda_{\max}^{\mathcal{A}}/\theta)$ and $\|\tilde{Q}^{-1}\| = \mathcal{O}([1+\lambda_{\min}^{\mathcal{A}}/\theta]^{-1})$. Third, in view of the latter two observations, the above procedure based on the CG method is more suitable for those instances such that: i) the amount of memory required to store $\mathcal{A}$, either explicitly or implicitly, is substantially smaller than the one required to store $\mathcal{A}\mathcal{A}^*$ and its Cholesky factorization; and ii) bound (3.2.4) multiplied by the arithmetic complexity of an iteration of the CG method is relatively smaller than the arithmetic complexity of directly solving the linear system $\tilde{\mathcal{Q}}\Delta_y^k = \tilde{q}^k$ (see O.3 in Section 1.1). Fourth, the bound in (3.2.4) does not depend on the iteration count k of Algorithm 3.1 for fixed θ (see Figure 3.2.1). Finally, the bound (3.2.4) is strictly decreasing as a function of θ.

3.2.2 Error measures and dynamic scaling

In this subsection, we describe three measures that quantify the optimality of an approximate solution of (1.1.1), namely: the primal infeasibility measure; the dual infeasibility measure; and the relative duality gap. We also describe two important refinements of Algorithm 3.1 based on the ideas introduced in [35] and [33]. More specifically, we describe: i) a scheme for choosing the initial scaling parameters θ and ξ; and ii) a procedure for

dynamically updating the scaling parameters θ and ξ to balance the sizes of three error measures as the algorithm progresses.

For the purpose of describing a stopping criterion for Algorithm 3.1, define the primal infeasibility measure as

$$\epsilon_P(x) := \frac{\|\mathcal{A}x - b\|}{\|b\| + 1} \qquad \forall x \in \mathcal{X}, \tag{3.2.5}$$

and the dual infeasibility measure as

$$\epsilon_D(y, z) := \frac{\|c - \mathcal{A}^* y - z\|}{\|c\| + 1} \qquad \forall (y, z) \in \mathcal{Y} \times \mathcal{X}. \tag{3.2.6}$$

Finally, define the relative duality gap as

$$\epsilon_G(x, y) := \frac{\langle c, x\rangle - \langle b, y\rangle}{|\langle c, x\rangle| + |\langle b, y\rangle| + 1} \qquad \forall x \in \mathcal{X},\ \forall y \in \mathcal{Y}. \tag{3.2.7}$$

For given tolerances $\bar{\epsilon} > 0$, we stop Algorithm 3.1 whenever

$$\max\{\epsilon_{P,k}, \epsilon_{D,k}, |\epsilon_{G,k}|\} \leq \bar{\epsilon}, \tag{3.2.8}$$

where

$$\epsilon_{P,k} := \epsilon_P(\Pi_K(\tilde{x}^k)), \qquad \epsilon_{D,k} := \epsilon_D(\tilde{y}^k, \tilde{z}^k), \qquad \epsilon_{G,k} := \epsilon_G(\Pi_K(\tilde{x}^k), \tilde{y}^k).$$

We now make some observations about the stopping criterion (3.2.8). First, the primal and dual infeasibility measures in (3.2.5) and (3.2.6) do not take into consideration violations with respect to the constraints $x \in K$ and $z \in K^*$, respectively. Since we evaluate them at $(x, y, z) = (\Pi_K(\tilde{x}^k), \tilde{y}^k, \tilde{z}^k)$ and in this case $(x, z) \in K \times K^*$, there is no need to take these violations into account. Second, from the definition of Π_K, Corollary 3.3(a), and identities (3.1.14b) and (3.1.14c), it follows that

$$\begin{aligned}
\epsilon_{P,k} &= \frac{\|r_y^k + \mathcal{A}(\Pi_K(\tilde{x}^k) - \tilde{x}^k)\|}{\|b\| + 1}, \qquad \epsilon_{D,k} = \frac{\|r_x^k\|}{\|c\| + 1}, \\
\epsilon_{G,k} &= \frac{\langle r_x^k, \tilde{x}^k\rangle + \langle r_y^k, \tilde{y}^k\rangle + \langle r_z^k, \tilde{z}^k\rangle + \langle c, \Pi_K(\tilde{x}^k) - x^k\rangle}{|\langle c, \Pi_K(\tilde{x}^k)\rangle| + |\langle b, \tilde{y}^k\rangle| + 1}, \\
\|\Pi_K(\tilde{x}^k) - x^k\| &\leq \|\tilde{u}^k - x^k\| = \|r_z^k\|,
\end{aligned}$$

which together with Theorem 3.2 imply that zero is a cluster value of the sequences $\{\epsilon_{P,k}\}$, $\{\epsilon_{D,k}\}$ and $\{\epsilon_{G,k}\}$ as $k \to \infty$. Hence, Algorithm 3.1 with the termination criterion (3.2.8)

will eventually terminate. Third, another possibility is to terminate Algorithm 3.1 based on the quantities $\epsilon'_{P,k} = \epsilon_P(\tilde{u}^k)$, $\epsilon_{D,k}$ and $\epsilon'_{G,k} := \epsilon_G(\tilde{u}^k, \tilde{y}^k)$, which also approach zero (in a cluster sense) due to Theorem 3.2 and Corollary 3.3. Our current implementation of Algorithm 3.1 ignores the latter possibility and terminates based on (3.2.8). Finally, it should be observed that the termination criterion (3.2.8) requires the evaluation of an additional projection for computing $\epsilon_{P,k}$ and $\epsilon_{G,k}$, namely, $\Pi_K(\tilde{x}^k)$. To avoid computing this projection at every iteration, our implementation of Algorithm 3.1 only checks whether (3.2.8) is satisfied when $\max\left\{\epsilon_P(\tilde{x}^k), \epsilon_{D,k}, |\epsilon_G(\tilde{x}^k, \tilde{y}^k)|\right\} \leq \bar{\epsilon}$ holds.

We now discuss two important refinements of Algorithm 3.1 whose goal is to balance the magnitudes of the scaled residuals

$$\rho_{z,k} := \frac{\left\|r_z^k\right\|}{|\langle c, \tilde{x}^k\rangle| + |\langle b, \tilde{y}^k\rangle| + 1}, \quad \rho_{y,k} := \frac{\left\|r_y^k\right\|}{\|b\| + 1}, \quad \rho_{x,k} := \frac{\left\|r_x^k\right\|}{\|c\| + 1}, \quad , \tag{3.2.9}$$

where r_z^k, r_y^k and r_x^k are defined in Theorem 3.2. Observe that (3.2.9) imply that $\mathcal{R}_{\theta,k} := \max\{\rho_{y,k}, \rho_{x,k}\}/\rho_{z,k} = \mathcal{O}\left(\max\{\|r_y^k\|, \|r_x^k\|\}/\|r_z^k\|\right)$ and $\mathcal{R}_{\xi,k} := \rho_{y,k}/\rho_{x,k} = \mathcal{O}\left(\|r_y^k\|/\|r_x^k\|\right)$. Hence, in view of the second observation in the paragraph following Theorem 3.2, the ratio $\mathcal{R}_{\theta,k}$ (resp., $\mathcal{R}_{\xi,k}$) can grow significantly as $\theta \to \infty$ (resp, $\xi \to \infty$), while it can become very small as $\theta \to 0$ (resp., $\xi \to 0$). This suggests that the ratio $\mathcal{R}_{\theta,k}$ (resp., $\mathcal{R}_{\xi,k}$) increases as θ (resp., ξ) increases, and decreases as θ (resp., ξ) decreases. Indeed, our computational experiments indicate that the ratios $\mathcal{R}_{\theta,k}$ and $\mathcal{R}_{\xi,k}$ behave in this manner.

In the following, let θ_k and ξ_k denote the dynamic values of θ and ξ at the kth iteration of Algorithm 3.1, respectively. Observe that, in view of (3.1.11), (3.1.13) and (3.2.9), the measures $\rho_{z,k}$, $\rho_{y,k}$ and $\rho_{x,k}$ depend on $\tilde{z}^k$, $\tilde{y}^k$ and $\tilde{x}^k$, whose values in turn depend on the choice of θ_k and ξ_k, in view of steps 1 and 2 of Algorithm 3.1. Hence, these measures are indeed functions of θ and ξ, which are denoted as$\rho_{z,k}(\theta, \xi)$, $\rho_{y,k}(\theta, \xi)$ and $\rho_{x,k}(\theta, \xi)$.

We first describe a scheme for choosing the initial scaling parameters θ_1 and ξ_1. Let a constant $\rho > 1$ be given and tentatively set $\theta = \xi = 1$. If $\rho_{y,1}(\theta, \xi)/\rho_{x,1}(\theta, \xi) > \rho$ (resp., $\rho_{y,1}(\theta, \xi)/\rho_{x,1}(\theta, \xi) < \rho^{-1}$), we successively divide (resp., successively multiply) the current value of ξ by 2 until $\rho_{y,1}(\theta, \xi)/\rho_{x,1}(\theta, \xi) \leq \rho$ (resp., $\rho_{y,1}(\theta, \xi)/\rho_{x,1}(\theta, \xi) \geq \rho^{-1}$) is satisfied, and set $\xi_1 = \xi_1^*$ where ξ_1^* is the last value of ξ. Since we have not updated θ,

at this stage we still have $\theta = 1$. At the second stage of this scheme, we update θ in exactly the same manner as above, but in place of $\rho_{y,1}(\theta,\xi)/\rho_{x,1}(\theta,\xi)$ we use the ratio $\max\{\rho_{y,1}(\theta,\xi),\rho_{x,1}(\theta,\xi)\}/\rho_{z,1}(\theta,\xi)$, and set $\theta_1 = \theta_1^*$ where θ_1^* is the last value of θ. Since there is no guarantee that the latter scheme will terminate, we specify an upper bound on the number of times that ξ and θ can be updated. In our implementation, we set this upper bound to be 20.

We next describe a procedure for dynamically updating the scaling parameters θ and ξ to balance the sizes of the measures $\rho_{z,k}(\theta,\xi)$, $\rho_{y,k}(\theta,\xi)$ and $\rho_{x,k}(\theta,\xi)$ as the algorithm progresses. Similar to the dynamic procedures used in [35] and [33], we use the heuristic of changing θ_k and ξ_k every time a specified number of iterations have been performed. More specifically, given an integer $\bar{k} \geq 1$, and scalars $\gamma_1, \gamma_2 > 1$ and $0 < \tau < 1$, if

$$\gamma_2 \exp\left(\left|\ln\left(\max\{\bar{\rho}_{y,k},\bar{\rho}_{x,k}\}/\bar{\rho}_{z,k}\right)\right|\right) > \gamma_1 \exp\left(\left|\ln\left(\bar{\rho}_{y,k}/\bar{\rho}_{x,k}\right)\right|\right)$$

we set $\xi_{k+1} = \xi_k$ and use the following dynamic scaling procedure for updating θ_{k+1},

$$\theta_{k+1} = \begin{cases} \theta_k, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma_1^{-1} \leq \max\{\bar{\rho}_{y,k},\bar{\rho}_{x,k}\}/\bar{\rho}_{z,k} \leq \gamma_1 \\ \tau^2\theta_k, & k \equiv 0 \bmod \bar{k} \text{ and } \max\{\bar{\rho}_{y,k},\bar{\rho}_{x,k}\}/\bar{\rho}_{z,k} > \gamma_1 \\ \tau^{-2}\theta_k, & k \equiv 0 \bmod \bar{k} \text{ and } \max\{\bar{\rho}_{y,k},\bar{\rho}_{x,k}\}/\bar{\rho}_{z,k} < \gamma_1^{-1} \end{cases} \quad \forall k \geq 1, \quad (3.2.10)$$

otherwise, we set $\theta_{k+1} = \theta_k$ and use the following dynamic scaling procedure for updating ξ_{k+1},

$$\xi_{k+1} = \begin{cases} \xi_k, & k \not\equiv 0 \bmod \bar{k} \text{ or } \gamma_2^{-1} \leq \bar{\rho}_{y,k}/\bar{\rho}_{x,k} \leq \gamma_2 \\ \tau^2\xi_k, & k \equiv 0 \bmod \bar{k} \text{ and } \bar{\rho}_{y,k}/\bar{\rho}_{x,k} > \gamma_2 \\ \tau^{-2}\xi_k, & k \equiv 0 \bmod \bar{k} \text{ and } \bar{\rho}_{y,k}/\bar{\rho}_{x,k} < \gamma_2^{-1} \end{cases} \quad \forall k \geq 1, \quad (3.2.11)$$

where

$$\bar{\rho}_{z,k} = \left(\prod_{i=k-\bar{k}+1}^{k} \rho_{z,i}\right)^{1/\bar{k}}, \qquad \bar{\rho}_{y,k} = \left(\prod_{i=k-\bar{k}+1}^{k} \rho_{y,i}\right)^{1/\bar{k}}, \qquad \bar{\rho}_{x,k} = \left(\prod_{i=k-\bar{k}+1}^{k} \rho_{x,i}\right)^{1/\bar{k}} \quad \forall k \geq \bar{k}. \quad (3.2.12)$$

In our computational experiments, we have used $\bar{k} = 10$, $\gamma_1 = 8$, $\gamma_2 = 2$ and $\tau = 0.9$. Roughly speaking, the above dynamic scaling procedure changes the values of θ and ξ at most a single time in the right direction, so as to balance the sizes of the residuals based on the information provided by their values at the previous $\bar{k}$ iterations. We observe that the above scheme is based on similar ideas as the ones used in Section 2.3, but involves two scaling parameters instead of only one as in Section 2.3. Also, since the bound on the CG iterations (3.2.4) increases as θ decreases, we stop decreasing θ whenever the CG method starts to perform relatively high number of iterations in order to obtain Δ_y^k satisfying (3.2.3).

In our computational experiments, we refer to the variant of Algorithm 3.1 in which incorporates the above dynamic scaling scheme and uses the CG method to perform step 2 as explained in Subsection 3.2.1 as the *CG based inexact scaled block-decomposition* (CG-ISBD) method. Figure 3.2.2 compares the performance of the CG-ISBD method on a conic SDP instance against the following three variants: i) VAR1, the one that removes the dynamic scaling (i.e., set $\theta_k = \theta_1^*$ and $\xi_k = \xi_1^*$, for every $k \geq 1$); ii) VAR2, the one that removes the dynamic scaling and the initialization scheme for θ_1 (i.e., set $\theta_k = 1$ and $\xi_k = 1$, for every $k \geq 1$); and iii) VAR3, the one that removes these latter two refinements and the use of adaptive stepsize (i.e., set $\theta_k = 1$, $\xi_k = 1$ and $\lambda_k = \tilde{\lambda} = \sqrt{\sigma^2 - \sigma_w^2}$, for every $k \geq 1$).

3.3 Numerical results

In this section, we compare the CG-ISBD method described in Section 3.2 with the SDPLR method discussed in [7, 8, 6]. More specifically, we compare these two methods on a collection of extra large-scale conic SDP instances of (1.1.1) and (1.1.4) where the size and/or density of the linear operator $\mathcal{A}$ is such that the operation of projecting a point onto the manifold $\mathcal{M}$ (see O.3 in Section 1.1) is prohibitively expensive.

We have implemented CG-ISBD for solving (1.1.1) in a MATLAB code which can be found at `http://www.isye.gatech.edu/~cod3/COrtiz/software/`. This variant was implemented for spaces $\mathcal{X}$ and $\mathcal{Y}$, and cone K given as in (1.1.4). Hence, our code is able to solve conic programming problems given in standard form (i.e., as in (1.1.1)) with n_u unrestricted scalar variables, n_l nonnegative scalar variables and an $n_s \times n_s$ positive semidefinite

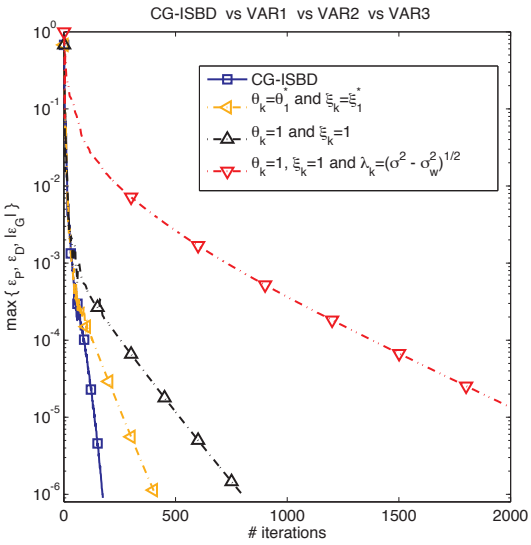

Figure 3.2.2: This example (random SDP instance) illustrates how all the refinements made in the application of the Special-BD framework to problem (1.1.1) helped improve the performance of the algorithm.

symmetric matrix variable. The inner products (before scaling) used in $\mathcal{X}$ and $\mathcal{Y}$ are the standard ones, namely: the scalar inner product in $\mathcal{Y}$ and the following inner product in $\mathcal{X}$

$$\langle x, \tilde{x} \rangle := x_v^T \tilde{x}_v + X_s \bullet \tilde{X}_s,$$

for every $x = (x_v, X_s) \in \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}$ and $\tilde{x} = (\tilde{x}_v, \tilde{X}_s) \in \mathbb{R}^{n_u+n_l} \times \mathcal{S}^{n_s}$, where $X \bullet Y := Tr(X^T Y)$ for every $X, Y \in \mathcal{S}^{n_s}$. Also, this implementation uses the preconditioned CG procedure `pcg.m` from MATLAB with a modified stopping criterion to obtain Δ_y^k as in Subsection 3.2.1. On the other hand, we have used the 1.03-beta C implementation of SDPLR[1]. All the computational results were obtained on a single core of a server with 2 Xeon E5-2630 processors at 2.30GHz and 64GB RAM.

In our benchmark, we have considered various large-scale random SDP (RAND) problems, and two large and dense SDP bounds of the Ramsey multiplicity problem (RMP). The RAND instances are pure SDPs, i.e., instances of (1.1.1) and (1.1.4) with $n_l = n_u = 0$, and were generated using the same random SDP generator used in [30]. In Appendix A.4, we describe in detail the RMP and the structure of its SDP bound. For each one of the

[1]Available at `http://dollar.biz.uiowa.edu/~sburer/files/SDPLR-1.03-beta.zip`.

above conic SDP instances, both methods stop whenever (2.3.44) with $\bar{\epsilon} = 10^{-6}$ is satisfied with an upper bound of 300,000 seconds running time.

We now make some general remarks about how the results are reported on the tables given below. Table 3.3.1 reports the size and the number of non-zeros of $\mathcal{A}$ and $\mathcal{A}\mathcal{A}^*$ for each conic SDP instance. All the instances are large and dense enough so that our server runs out of memory when trying to compute a projection onto the manifold $\mathcal{M}$ (see O.3 in Section 1.1) based on the Cholesky factorization of $\mathcal{A}\mathcal{A}^*$. Table 3.3.2, which compares CG-ISBD against SDPLR, reports the primal and dual infeasibility measures as described in (2.3.41) and (2.3.42), respectively, the relative duality gap as in (2.3.43) and the time (in seconds) for both methods at the last iteration. In addition, Table 3.3.2 includes the number of (outer) iterations and the total number of CG iterations performed by CG-ISBD. The residuals (i.e., the primal and dual infeasibility measures, and the relative duality gap) and time taken by any of the two methods for any particular instance are marked in red, and also with an asterisk (*), whenever it cannot solve the instance by the required accuracy, in which case the residuals reported are the ones obtained at the last iteration of the method. Moreover, the time is marked with two asterisks (**) whenever the method is stopped due to numerical errors (e.g., NaN values are obtained). Also, the fastest time in each row is marked in bold.

Finally, Figure 3.3.1 plots the performance profiles (see [15]) of CG-ISBD and SDPLR methods based on all instances used in our benchmark.

Based on Table 3.3.2 and the performance profiles in Figure 3.3.1 and the remarks that follow, we can conclude that CG-ISBD substantially outperforms SDPLR in this benchmark. First, CG-ISBD is able to solve all of the test instances with accuracy $\bar{\epsilon} \leq 10^{-6}$ faster than SDPLR, even though SDPLR fails to obtain a solution with such accuracy for 50% of them. Second, the performance profiles in Figure 3.3.1 show that SDPLR takes at least 4 and sometimes as much as 100 times more running time than CG-ISBD for almost all of the instances that it was able to solve with accuracy $\bar{\epsilon} \leq 10^{-6}$. Third, CG-ISBD is able to solve the two largest problems (`RAND_3k_1.9M_p4` and `RAND_3k_2M_p4`) in terms of number of constraints in approximately one tenth of the maximum running time allocated

Table 3.3.1: Extra large-scale conic SDP test instances.

Problem		Sparsity	
INSTANCE	$n_s; n_l \vert m$	nnz($\mathcal{A}$)	nnz($\mathcal{A}\mathcal{A}^*$)
RAND_n1K_m200K_p4	1000; 0|200000	1,194,036	3,050,786
RAND_n1.5K_m250K_p4	1500; 0|200000	1,492,374	2,225,686
RAND_n1.5K_m400K_p4	1500; 0|250000	2,387,989	5,462,130
RAND_n1.5K_m400K_p5	1500; 0|400000	3,980,228	14,444,644
RAND_n1.5K_m500K_p4	1500; 0|400000	2,984,924	8,409,200
RAND_n1.5K_m500K_p5	1500; 0|500000	4,974,944	22,420,950
RAND_n1.5K_m600K_p4	1500; 0|500000	3,582,041	11,989,112
RAND_n1.5K_m600K_p5	1500; 0|600000	5,970,139	32,168,670
RAND_n1.5K_m700K_p4	1500; 0|600000	4,179,023	16,202,622
RAND_n1.5K_m800K_p4	1500; 0|700000	4,775,861	21,042,204
RAND_n1.5K_m800K_p5	1500; 0|800000	7,959,941	56,937,526
RAND_n2K_m250K_p4	1500; 0|800000	1,492,475	1,364,154
RAND_n2K_m300K_p4	2000; 0|250000	1,790,965	1,899,818
RAND_n2K_m600K_p4	2000; 0|300000	3,582,095	7,011,000
RAND_n2K_m600K_p5	2000; 0|600000	5,970,069	18,367,654
RAND_n2K_m700K_p4	2000; 0|600000	4,179,126	9,428,520
RAND_n2K_m700K_p5	2000; 0|700000	6,965,068	24,889,158
RAND_n2K_m800K_p4	2000; 0|700000	4,776,139	12,199,146
RAND_n2K_m800K_p5	2000; 0|800000	7,959,854	32,408,642
RAND_n2K_m900K_p4	2000; 0|800000	5,372,860	15,328,086
RAND_n2K_m1M_p4	2000; 0|900000	5,969,926	18,804,856
RAND_n2.5K_m1.4M_p4	2000; 0|1000000	8,357,945	23,740,864
RAND_n2.5K_m1.5M_p4	2500; 0|1400000	8,954,920	27,140,896
RAND_n2.5K_m1.6M_p4	2500; 0|1500000	9,551,974	30,769,976
RAND_n2.5K_m1.7M_p4	2500; 0|1600000	10,148,881	34,633,858
RAND_n2.5K_m1.8M_p4	2500; 0|1700000	10,746,281	38,718,638
RAND_n3K_m1.6M_p4	2500; 0|1800000	9,551,975	21,863,148
RAND_n3K_m1.7M_p4	3000; 0|1600000	10,149,092	24,581,412
RAND_n3K_m1.8M_p4	3000; 0|1700000	10,745,808	27,453,710
RAND_n3K_m1.9M_p4	3000; 0|1800000	11,343,196	30,485,510
RAND_n3K_m2M_p4	3000; 0|1900000	11,939,933	33,663,958
RMP_1	3000; 0|2000000	53,986,254	75,442,510,224
RMP_2	90; 274668|274668	53,986,254	75,442,510,224

Table 3.3.2: Extra large SDP instances solved using SDPLR and CG-ISBD with accuracy $\bar{\varepsilon} \leq 10^{-6}$.

Problem	Error Measures						Performance			
	CG-ISBD			SDPLR			CG-ISBD			SDPLR
INSTANCE	ϵ_P	ϵ_D	ϵ_G	ϵ_P	ϵ_D	ϵ_G	OUT-IT	CG-IT	TIME	TIME
RAND_n1K_m200K_p4	9.96 -7	4.33 -8	+6.62 -7	5.92 -12	9.98 -7	+6.37 -7	343	20778	**1736**	186308
RAND_n1.5K_m200K_p4	9.77 -7	7.23 -9	+8.41 -7	1.07 -8	8.18 -7	+8.17 -7	452	18043	**4330**	16466
RAND_n1.5K_m250K_p4	3.17 -7	2.09 -9	+9.51 -7	1.37 -9	4.38 -7	+4.12 -7	448	20050	**4880**	28980
RAND_n1.5K_m400K_p4	9.83 -7	2.62 -8	+2.66 -7	1.48 -9	9.15 -7	-1.80 -7	365	20932	**6309**	39937
RAND_n1.5K_m400K_p5	8.33 -7	1.66 -8	+9.13 -7	3.80 -10	3.47 -7	+9.73 -7	366	14108	**5978**	30997
RAND_n1.5K_m500K_p4	1.44 -7	3.10 -9	+9.60 -7	8.77 -12	1.77 -7	+7.38 -6*	348	23895	**6880**	300000*
RAND_n1.5K_m500K_p5	2.41 -7	5.57 -9	+9.44 -7	3.38 -10	1.99 -7	-8.92 -8	338	15707	**7120**	34877
RAND_n1.5K_m600K_p4	9.39 -7	4.79 -8	+3.32 -7	1.11 -11	9.62 -7	+2.35 -6*	281	22408	**8004**	284521**
RAND_n1.5K_m600K_p5	9.67 -7	3.84 -8	+8.01 -8	4.85 -12	1.02 -6*	+2.22 -6*	285	15844	**8260**	266376**
RAND_n1.5K_m700K_p4	9.86 -7	2.18 -9	+6.17 -9	2.80 -11	2.78 -8	-1.42 -6*	284	26690	**10987**	300000*
RAND_n1.5K_m800K_p4	2.77 -7	3.90 -9	+6.35 -7	1.42 -8	6.22 -8	+1.08 -7	291	39274	**17508**	19056
RAND_n1.5K_m800K_p5	9.74 -7	1.27 -8	+7.40 -7	1.34 -9	3.08 -8	-5.43 -7	275	25877	**14463**	37926
RAND_n2K_m250K_p4	9.87 -7	4.81 -8	+3.00 -7	1.84 -10	7.75 -7	+3.28 -6*	292	20743	**14362**	300000*
RAND_n2K_m300K_p4	9.37 -7	3.45 -9	+3.06 -7	9.35 -10	3.75 -6*	-9.58 -7	480	17391	**8348**	300000*
RAND_n2K_m600K_p4	9.93 -7	4.84 -9	+3.04 -7	1.99 -10	3.06 -7	+7.60 -7	464	20193	**10433**	228621
RAND_n2K_m600K_p5	9.84 -7	1.72 -8	+3.31 -8	6.72 -9	8.64 -7	-3.30 -7	390	21856	**11612**	51172
RAND_n2K_m700K_p4	9.48 -7	1.27 -8	+8.72 -7	6.63 -10	3.16 -7	+9.76 -7	388	13654	**10657**	51347
RAND_n2K_m700K_p5	9.50 -7	2.12 -8	+9.05 -7	1.06 -10	6.87 -7	+4.12 -6*	369	21296	**11033**	300000*
RAND_n2K_m800K_p4	9.58 -7	1.98 -8	+7.94 -7	8.23 -11	5.67 -7	+2.32 -6*	359	13712	**11442**	300000*
RAND_n2K_m800K_p5	6.72 -7	1.79 -8	+9.77 -7	3.22 -10	9.11 -7	+3.16 -6*	349	23564	**13943**	300000*
RAND_n2K_m900K_p4	5.34 -7	1.13 -8	+9.82 -7	6.94 -10	4.00 -7	+2.15 -7	347	14710	**12273**	98772
RAND_n2K_m1M_p4	9.62 -7	1.80 -8	+3.64 -6	4.23 -10	9.22 -7	+8.59 -6*	329	23185	**14229**	300000*
RAND_n2.5K_m1.4M_p4	6.67 -7	1.96 -8	+9.89 -7	5.14 -9	7.97 -7	-3.91 -7	325	22954	**22746**	184138
RAND_n2.5K_m1.5M_p4	9.39 -7	4.01 -8	+7.77 -7	1.75 -8	8.78 -7	+6.78 -7	300	22011	**22572**	89482
RAND_n2.5K_m1.6M_p4	9.83 -7	4.15 -8	+2.16 -7	5.49 -10	4.61 -7	+2.05 -6*	290	22605	**24094**	300000*
RAND_n2.5K_m1.7M_p4	7.71 -7	2.52 -8	+9.99 -7	5.57 -9	5.90 -7	+5.26 -8	283	25173	**29111**	131184
RAND_n2.5K_m1.8M_p4	9.78 -7	2.74 -8	+7.77 -7	2.88 -9	5.75 -7	+7.37 -7	271	24614	**27274**	195103
RAND_n3K_m1.6M_p4	9.78 -7	1.81 -8	+3.30 -7	1.17 -8	9.19 -7	+2.72 -6*	365	21464	**29711**	300000*
RAND_n3K_m1.7M_p4	7.94 -7	1.59 -8	+8.96 -7	6.48 -9	4.80 -7	+7.35 -6*	359	27262	**35973**	300000*
RAND_n3K_m1.8M_p4	9.72 -7	1.36 -8	+2.36 -6	9.01 -9	6.44 -7	+4.11 -7	351	22780	**32586**	211309
RAND_3k_1.9M_p4	3.08 -7	5.52 -9	+9.63 -7	1.31 -9	4.73 -7	-4.07 -6*	352	26610	**37020**	300000*
RAND_3k_2M_p4	7.49 -7	2.29 -8	+9.16 -7	6.06 -9	1.07 -6*	+7.38 -6*	322	23200	**33294**	300000*
RMP_1	7.96 -7	7.36 -7	+5.93 -7	3.22 -11	1.98 +0*	+6.60 -2*	5899	204632	**158023**	300000*
RMP_2	4.61 -7	9.49 -9	+3.76 -9	6.78 -13	2.06 +1*	-2.39 -3*	1034	58910	**52698**	164626**

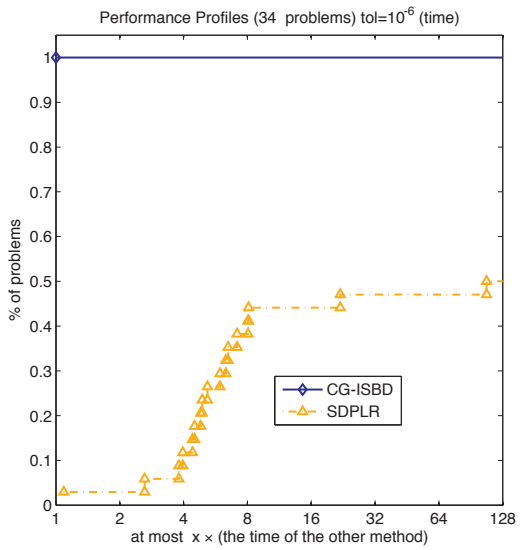

Figure 3.3.1: Performance profiles of CG-ISBD and SDPLR for solving 34 extra large-scale conic SDP instances with accuracy $\bar{\epsilon} = 10^{-6}$.

to SDPLR, which in turn was not able to solve them. Finally, CG-ISBD is able to solve the extremely dense instances `RMP_1` and `RMP_2` with accuracy 10^{-6} on all the residuals, but SDPLR significantly failed to decrease the dual infeasibility measures and relative duality gaps for both problems.

3.4 Concluding remarks

We now make a few remarks about the method CG-ISBD proposed in this chapter. First, the CG-ISBD method is able to avoid the operation of projecting onto the manifold $\mathcal{M}$ by iterating a CG subroutine until (2.1.6) of step 2 in the Special BD framework is satisfied (see Subsection 3.2.1). However, an alternative (more aggressive) possibility is to iterate the CG subroutine until the HPE condition (2.1.7) is satisfied with $\lambda = \tilde{\lambda}$. Second, it should be noted that the goal of the dynamic scaling scheme described in Subsection 3.2.2 is to reduce the number of outer iterations performed by CG-ISBD, and does not take into account the possibility of also reducing the number of inner iterations performed by the CG subroutine. The development of dynamic scaling schemes which take into account both the outer and inner iterations in order to improve the overall performance of the method is a topic for

further study.

Acknowledgments. We want to thank Susanne Nieß for generously providing us with the code and data sets to generate SDP bounds of RMPs as in [49].

Chapter IV

ADAPTIVE ACCELERATED GRADIENT METHODS FOR CONVEX OPTIMIZATION

In this chapter, we present an adaptive accelerated gradient method for convex optimization problems. This chapter is divided in five sections. First, we present a generic accelerated framework for convex optimization (1.2.1), namely the GenA framework, in Section 4.1. Section 4.2 introduces a scheme that adaptively and aggressively chooses certain acceleration parameters of the GenA framework in order to substantially improve its practical performance without compromising its theoretical iteration-complexity. Section 4.3 describes a practical first-order instance of the GenA framework that incorporates the latter scheme. In Section 4.4 we present numerical results of two implementations of the method of Section 4.3 that use adaptive restart schemes, demonstrating that they perform quite well compared to other variants of Nesterov's method proposed earlier in the literature. Finally, Section 4.5 presents some final remarks.

4.1 A generic framework of accelerated methods

In this section, we discuss an accelerated framework for convex optimization problems (1.2.1). This framework can be viewed as a generalization of the accelerated HPE (A-HPE) framework discussed in [38] with two main modifications: i) it considers a different reformulation of the main convergence condition; and ii) it imposes a weaker condition on the aggregated stepsizes.

We now state a generic framework of accelerated methods for solving (1.2.1).

GenA Framework: A generic accelerated framework

0) Let $\tau \in [0,1]$, $A_0 \in \mathbb{R}_+$, $y^0, u^0 \in \mathcal{X}$ and an affine function $\Gamma_0 : \mathcal{X} \to \mathbb{R}$ such that

$$A_0\Gamma_0 \leq A_0\phi, \quad A_0\phi(y^0) \leq \inf_{x\in\mathcal{X}} \left\{ A_0\Gamma_0(x) + \frac{1}{2}\|x - u^0\|^2 \right\}, \tag{4.1.1}$$

be given and set $k = 0$;

1) set

$$x^k = u^0 - A_k\nabla\Gamma_k = \arg\min_{x\in\mathcal{X}} \left\{ A_k\Gamma_k(x) + \frac{1}{2}\|x - u^0\|^2 \right\}; \tag{4.1.2}$$

2) compute a stepsize $\lambda_{k+1} > 0$, a point $\tilde{y}^{k+1} \in \mathcal{X}$, and an affine function $\gamma_{k+1} : \mathcal{X} \to \mathbb{R}$ minorizing ϕ such that

$$\lambda_{k+1}\phi(\tilde{y}^{k+1}) + \frac{1}{2}(1-\tau)\|\tilde{y}^{k+1} - \tilde{x}^k\|^2 \leq \min_{x\in\mathcal{X}} \left\{ \lambda_{k+1}\gamma_{k+1}(x) + \frac{1}{2}\|x - \tilde{x}^k\|^2 \right\}, \tag{4.1.3}$$

where

$$\tilde{x}^k = \frac{A_k}{A_k + a_{k+1}}\, y^k + \frac{a_{k+1}}{A_k + a_{k+1}}\, x^k, \tag{4.1.4}$$

$$a_{k+1} = \frac{\lambda_{k+1} + \sqrt{\lambda_{k+1}^2 + 4\lambda_{k+1}A_k}}{2}; \tag{4.1.5}$$

3) choose $y^{k+1} \in \mathcal{X}$ and the pair $(A'_k, a'_{k+1}) \in \mathbb{R}^2$ such that $\phi(y^{k+1}) \leq \phi(\tilde{y}^{k+1})$ and

$$A'_k \geq 0, a'_{k+1} \geq 0, \quad A'_k + a'_{k+1} \geq A_k + a_{k+1}, \tag{4.1.6}$$

$$(A'_k + a'_{k+1})\phi(y^{k+1}) \leq \inf_{x\in\mathcal{X}} \left\{ (A'_k\Gamma_k + a'_{k+1}\gamma_{k+1})(x) + \frac{1}{2}\|x - u^0\|^2 \right\}, \tag{4.1.7}$$

with the safeguard that $A'_0 = A_0$ when $k = 0$;

4) set

$$A_{k+1} := A'_k + a'_{k+1}, \quad \Gamma_{k+1} := \frac{A'_k}{A'_k + a'_{k+1}}\Gamma_k + \frac{a'_{k+1}}{A'_k + a'_{k+1}}\gamma_{k+1}, \tag{4.1.8}$$

$k \leftarrow k + 1$, and go to step 1.

We now make some remarks about the GenA framework. First, for every $k \geq 0$ and

$\lambda_{k+1} > 0$, the pair $(\tilde{y}^{k+1}, \gamma_{k+1}) = (\tilde{y}^E, \gamma_E)$, where

$$\tilde{y}^E := (I + \lambda_{k+1}\partial\phi)^{-1}(\tilde{x}^k), \qquad \gamma_E(x) = \phi(\tilde{y}^E) + \frac{1}{\lambda_{k+1}}\langle x - \tilde{y}^E, \tilde{x}^k - \tilde{y}^E\rangle,$$

can be easily seen to satisfy (4.1.3) (see Lemma 4.9). Hence, it is always possible to choose λ_{k+1}, $\tilde{y}^{k+1}$ and γ_{k+1} as in step 2 of the GenA framework. However, the above choice is not practical since it requires *exact* computation of the resolvent $(I + \lambda_{k+1}\partial\phi)^{-1}$ of ϕ. We will discuss in Section 4.3 more practical ways of computing the above three entities so as to satisfy the requirements of step 2. Second, if no pair (A_0, Γ_0) as in step 0 such that $A_0 > 0$, and hence $\Gamma_0 \leq \phi$, is known, then we can always choose $A_0 = 0$ and $\Gamma_0 \equiv 0$.

Before discussing the convergence properties of the GenA framework, we discuss the connection between the GenA framework stated here and the A-HPE framework presented in [38]. We first state two simple results.

Lemma 4.1. *Let $\chi \in \mathbb{R}$, $u^0, y \in \mathcal{X}$ and an affine function $\Gamma : \mathcal{X} \to \mathbb{R}$ be given. Then, the following conditions are equivalent:*

a) $\min\{\Gamma(x) + \|x - u^0\|^2/2 : x \in \mathcal{X}\} \geq \chi$;

b) $\chi - \Gamma(u^0) + \|\nabla\Gamma\|^2/2 \leq 0$;

c) $\|\nabla\Gamma + y - u^0\|^2 + 2[\chi - \Gamma(y)] \leq \|y - u^0\|^2$.

Proof. The result follows from the optimality conditions of the unconstrained optimization problem in a). □

Lemma 4.2. *Let $\lambda > 0$, $\tau \in [0, 1]$, $\tilde{x}, \tilde{y} \in \mathcal{X}$ be given. For any $\gamma : \mathcal{X} \to \mathbb{R}$ affine linear functional the following conditions are equivalent:*

a) $\gamma \leq \phi$ and

$$\lambda\phi(\tilde{y}) + \frac{1-\tau}{2}\|\tilde{y} - \tilde{x}\|^2 \leq \inf_{x\in\mathcal{X}}\left\{\lambda\gamma(x) + \frac{1}{2}\|x - \tilde{x}\|^2\right\}; \tag{4.1.9}$$

b) $\gamma(x) = \phi(\tilde{y}) + \langle v, x - \tilde{y}\rangle - \varepsilon$,

$$v \in \partial_\varepsilon\phi(\tilde{y}),\ \|\lambda v + \tilde{y} - \tilde{x}\|^2 + 2\lambda\varepsilon \leq \tau\|\tilde{y} - \tilde{x}\|^2 \tag{4.1.10}$$

Proof. To prove the equivalence of a) and b) first observe that if $\phi(\tilde{y}) = \infty$ neither a) nor b) hold. Now assume that $\phi(\tilde{y}) \in \mathbb{R}$. Under this assumption, any given affine functional γ has an (unique) expression as

$$\gamma(x) = \phi(\tilde{y}) + \langle v, x - \tilde{y} \rangle - \varepsilon$$

and, conversely, any $v \in \mathcal{X}$, $\varepsilon \in \mathbb{R}$ defines an (unique) affine functional by the above expression. In both cases, $v = \nabla\gamma$, $\varepsilon = \phi(\tilde{y}) - \gamma(\tilde{y})$ and this γ minorizes ϕ if and only is $v \in \partial_\varepsilon \phi(\tilde{y})$. Equivalence between (4.1.9) and (4.1.10) follows from the equivalence between a) and c) of Lemma 4.1 with $\Gamma = \lambda\gamma$, $u^0 = \tilde{x}$, $y = \tilde{y}$ and $\chi = \lambda\phi(\tilde{y}) + (1-\tau)\|\tilde{y} - u^0\|^2/2$. □

Hence, step 2) is equivalent to finding a triple $(\tilde{y}, v, \varepsilon) = (\tilde{y}^{k+1}, \nabla\gamma_{k+1}, \varepsilon_{k+1}) \in \mathcal{X} \times \mathcal{X} \times \mathbb{R}_+$ satisfying the error condition in Lemma 4.2(b):

$$v^{k+1} \in \partial_{\varepsilon_{k+1}} \phi(\tilde{y}^{k+1}), \quad \|\lambda_{k+1} v^{k+1} + \tilde{y}^{k+1} - \tilde{x}^k\|^2 + 2\lambda_{k+1}\varepsilon_{k+1} \leq \tau\|\tilde{y}^{k+1} - \tilde{x}^k\|^2. \quad (4.1.11)$$

Rather than using condition a) of Lemma 4.2 as in (4.1.3), the A-HPE framework studied in [38] uses the error condition in Lemma 4.2(b) as in (4.1.11). It can be easily shown that the GenA framework presented in this thesis with $(A'_k, a'_{k+1}) = (A_k, a_{k+1})$ is the same as the one studied in [38].

We now turn our attention to derive a bound on the functional gap $\phi(y^k) - \phi^*$, in terms of the aggregated acceleration parameter A_k, which is further refined to yield a bound in terms of the sequence of stepsizes $\{\lambda_k\}$.

We start by stating the following technical result.

Lemma 4.3. *Let $A \geq 0$, $u^0 \in \mathcal{X}$ and a closed convex function $\Gamma : \mathcal{X} \to \mathbb{R}$ be given and let*

$$x = \arg\min_{u \in \mathcal{X}} \left\{ A\Gamma(u) + \frac{1}{2}\|u - u^0\|^2 \right\}, \qquad \Theta = \min_{u \in \mathcal{X}} \left\{ A\Gamma(u) + \frac{1}{2}\|u - u^0\|^2 \right\}. \quad (4.1.12)$$

Let $\lambda > 0$, $y \in \mathcal{X}$ and a proper closed convex function $\gamma : \mathcal{X} \to \mathbb{R}$ minorizing ϕ be given, and define

$$a := \frac{\lambda + \sqrt{\lambda^2 + 4\lambda A}}{2}, \qquad \tilde{x} := \frac{A}{A+a} y + \frac{a}{A+a} x, \quad (4.1.13)$$

and

$$\theta := \min_{u \in \mathcal{X}} \left\{ \lambda\gamma(u) + \frac{1}{2}\|u - \tilde{x}\|^2 \right\}. \quad (4.1.14)$$

Then,

$$\min_{u\in\mathcal{X}}\left\{a\gamma(u)+A\Gamma(u)+\frac{1}{2}\|u-u^0\|^2\right\}\geq[\Theta-A\phi(y)]+(A+a)\frac{\theta}{\lambda}.$$

Proof. Let an arbitrary $u\in\mathcal{X}$ be given. Define

$$\tilde{u}:=\frac{Ay+au}{A+a}$$

and note that

$$\tilde{u}-\tilde{x}=\left(\frac{a}{A+a}\right)^2(u-x).$$

Note also that the definition of a in (4.1.13) implies that

$$\lambda=\frac{a^2}{A+a}.$$

In view of the two relations in (4.1.12), the fact that $\gamma\leq\phi$,the functions Γ and γ are convex, and the last three relations , we conclude that

$$\begin{aligned}
&a\gamma(u)+A\Gamma(u)+\frac{1}{2}\|u-u^0\|^2\\
&\quad\geq a\gamma(u)+\frac{1}{2}\|u-x\|^2+\Theta\\
&\quad\geq a\gamma(u)+\frac{1}{2}\|u-x\|^2+A\gamma(y)+[\Theta-A\phi(y)]\\
&\quad\geq(A+a)\gamma\left(\frac{Ay+au}{A+a}\right)+\frac{1}{2}\|u-x\|^2+[\Theta-A\phi(y)]\\
&\quad=(A+a)\left[\gamma(\tilde{u})+\frac{(A+a)}{2a^2}\|\tilde{u}-\tilde{x}\|^2\right]+[\Theta-A\phi(y)]\\
&\quad\geq(A+a)\min_{u'\in\mathcal{X}}\left\{\gamma(u')+\frac{1}{2\lambda}\|u'-\tilde{x}\|^2\right\}+[\Theta-A\phi(y)].
\end{aligned}$$

The result now follows from (4.1.14) and the fact that the above inequality holds for any $u\in\mathcal{X}$. □

We now use the above result to establish the following result.

Lemma 4.4. *The following statements hold for every $k\geq 0$:*

a) Γ_k is affine and $A_k\Gamma_k\leq A_k\phi$;

b) the pair $(A'_k,a'_{k+1})=(A_k,a_{k+1})$ satisfies (4.1.7)*;*

c) there holds

$$A_k\phi(y^k) \le \min_{u\in\mathcal{X}} \left\{ A_k\Gamma_k(u) + \frac{1}{2}\|u-u^0\|^2 \right\};$$

Proof. We first claim that a) and c) imply b). Indeed, assume that a) and c) hold. Using c) and Lemma 4.3 with $\Gamma = \Gamma_k$, $\gamma = \gamma_{k+1}$, $\lambda = \lambda_{k+1}$, $y = y^k$, $A = A_k$, we conclude that

$$\begin{aligned}\min_{u\in\mathcal{X}} \left\{ a_{k+1}\gamma_{k+1}(u) + A_k\Gamma_k(u) + \frac{1}{2}\|u-u^0\|^2 \right\} &\ge \frac{A_k+a_{k+1}}{\lambda_{k+1}} \min_{u\in\mathcal{X}} \left\{ \lambda_{k+1}\gamma_{k+1}(u) + \frac{1}{2}\|u-\tilde{x}^k\|^2 \right\} \\ &\ge (A_k+a_{k+1})\phi(\tilde{y}^{k+1}) \ge (A_k+a_{k+1})\phi(y^{k+1}),\end{aligned}$$

where the last two inequalities are due to (4.1.3) and step 3 of the GenA framework, respectively. We have thus shown that b) holds.

In view of the above claim, it suffices to show that a) and c) holds for every $k \ge 0$. The proof is by induction for k. They certainly hold for $k = 0$ due to (4.1.1) and the assumption that Γ_0 is affine. Assume now that a) and c) hold for k. By the above claim, b) also holds for k. This implies that k-th step of the GenA framework is well-defined, and hence that c) holds for $k+1$, due to (4.1.7) and the first identity in (4.1.8). Also, statement a) for k, relation (4.1.8), and the fact that γ_{k+1} is an affine function, imply that Γ_{k+1} is affine and

$$A_{k+1}\Gamma_{k+1} = A'_k\Gamma_k + a'_{k+1}\gamma_{k+1} \le A'_k\phi + a'_{k+1}\phi = A_{k+1}\phi,$$

thereby showing that a) holds for $k+1$. Thus, the result follows. □

The main result about the GenA framework is as follows.

Theorem 4.5. *Let x^* be the projection of u^0 onto X^* and d_0 be the distance of u^0 to X^*. Then, for every $k \ge 1$,*

$$\frac{1}{2}\|x^k - x^*\|^2 + A_k\left[\phi(y^k) - \phi^*\right] \le \frac{1}{2}\, d_0^2. \tag{4.1.15}$$

and

$$A_k \ge \frac{1}{4}\left(\sum_{j=1}^{k}\sqrt{\lambda_j}\right)^2. \tag{4.1.16}$$

As a consequence, for every integer $k \ge 1$,

$$\phi(y^k) - \phi^* \le \frac{d_0^2}{2A_k} \le 2d_0^2\left(\sum_{j=1}^{k}\sqrt{\lambda_j}\right)^{-2}, \qquad \|x^k - x^*\| \le d_0. \tag{4.1.17}$$

Proof. The fact that Γ_k is an affine function minorizing ϕ, the definition of x^* and (4.1.2) imply that

$$A_k\phi^* + \frac{1}{2}d_0^2 = A_k\Gamma_k(x^*) + \frac{1}{2}\|x^* - u^0\|^2 = \min_{x\in\mathcal{X}}\left\{A_k\Gamma_k(x) + \frac{1}{2}\|x - u^0\|^2\right\} + \frac{1}{2}\|x^* - x^k\|^2$$
$$\geq A_k\phi(y^k) + \frac{1}{2}\|x^* - x^k\|^2,$$

where the last inequality is due to Lemma 4.4(c). Hence, (4.1.15) follows. Now, by (4.1.5), we have

$$a_{k+1} \geq \frac{\lambda_{k+1}}{2} + \sqrt{\lambda_{k+1}A_k},$$

which, together with (4.1.6) and (4.1.8), imply that

$$A_{k+1} = A'_k + a'_{k+1} \geq A_k + a_{k+1} \geq A_k + \left(\frac{\lambda_{k+1}}{2} + \sqrt{\lambda_{k+1}A_k}\right) \geq \left(\sqrt{A_k} + \frac{1}{2}\sqrt{\lambda_{k+1}}\right)^2.$$

This clearly implies that

$$\sqrt{A_k} - \sqrt{A_0} \geq \frac{1}{2}\left(\sum_{j=1}^{k}\sqrt{\lambda_j}\right),$$

and hence that (4.1.16) holds. The two inequalities in (4.1.17) follow immediately from (4.1.15) and (4.1.16). □

4.2 Optimizing the aggregated stepsizes

In this section, we introduce a scheme that adaptively and aggressively chooses certain acceleration parameters of the GenA framework described in Section 4.1 in order substantially improve its practical performance. More specifically, in view of the update formula for A_{k+1} in the first relation in (4.1.8) of the GenA framework, and motivated by first inequality of the iteration-complexity bound given in (4.1.17), we introduce a scheme that aims to maximize the sum $A'_k + a'_{k+1}$ at every iteration.

The following result shows an alternative characterization for condition (4.1.7).

Proposition 4.6. *For every $k \geq 0$, condition* (4.1.7) *holds if, and only if,*

$$A'_k[\phi(y^{k+1}) - \Gamma_k(u^0)] + a'_{k+1}[\phi(y^{k+1}) - \gamma_{k+1}(u^0)] + \frac{1}{2}\|A'_k\nabla\Gamma_k + a'_{k+1}\nabla\gamma_{k+1}\|^2 \leq 0. \qquad (4.2.1)$$

Proof. This result follows from the equivalence between a) and b) of Lemma 4.1 with $\Gamma = A'_k\Gamma_k + a'_{k+1}\gamma_{k+1}$, $\chi = (A'_k + a'_{k+1})\phi(y^{k+1})$ and $y = y^{k+1}$. □

Note that (4.2.1) is a single convex quadratic constraint on the scalars A'_k and a'_{k+1}. In view of the first inequality in (4.1.17), which suggests that the rate of convergence of $\{\phi(y^k) - \phi^*\}$ to zero is faster the larger A_k is chosen, it seems natural to choose the pair (A'_k, a'_{k+1}) as the optimal solution of the two-variable convex quadratic constrained problem

$$\begin{aligned} &\max\ A' + a' \\ &\text{s.t. } A'_k[\phi(y^{k+1}) - \Gamma_k(u^0)] + a'_{k+1}[\phi(y^{k+1}) - \gamma_{k+1}(u^0)] + \frac{1}{2}\|A'_k\nabla\Gamma_k + a'_{k+1}\nabla\gamma_{k+1}\|^2 \le 0, \\ &\qquad A', a' \ge 0. \end{aligned} \tag{4.2.2}$$

Since (4.2.2) contains only two variables, an optimal solution of (4.2.2), when it exists (see Proposition 4.7 below), can be easily computed.

In the remaining part of this section, we discuss the case when problem (4.2.2) is unbounded. We will see that this case implies that $y^{k+1} \in X^*$, in which case we may successfully stop the GenA framework. In fact, we consider a slightly more general problem which covers problem (4.2.2) as a special case.

Proposition 4.7. *Suppose that $u^0, y \in \mathcal{X}$ and $\gamma_1, \ldots, \gamma_m : \mathcal{X} \to \mathbb{R}$ are affine functions minorizing ϕ and consider the problem*

$$\begin{aligned} &\max \quad \textstyle\sum_{i=1}^m \alpha_i \\ &\text{s.t.} \quad \inf_{x\in\mathcal{X}} \left\{\sum_{i=1}^m \alpha_i\gamma_i(x) + \frac{1}{2}\|x - u^0\|^2\right\} \ge \sum_{i=1}^m \alpha_i\phi(y), \\ &\qquad \alpha_1 \ge 0, \ldots, \alpha_m \ge 0. \end{aligned} \tag{4.2.3}$$

Then, the following conditions are equivalent:

a) problem (4.2.3) *is unbounded;*

b) there exist not all zero nonnegative scalars $\bar\alpha_1, \ldots, \bar\alpha_m$ such that

$$\sum_{i=1}^m \bar\alpha_i\nabla\gamma_i = 0, \qquad \bar\alpha_i[\phi(y) - \gamma_i(y)] = 0, \quad i = 1, \ldots, m. \tag{4.2.4}$$

In both cases, $y \in X^*$.

Proof. First, note that the equivalence between a) and c) of Lemma 4.1 with $\Gamma = \sum_{i=1}^m \alpha_i\gamma_i$ and $\chi = \sum_{i=1}^m \alpha_i\phi(y)$ imply that the hard constraint in problem (4.2.3) is equivalent to

$$\left\|\sum_{i=1}^m \alpha_i\nabla\gamma_i + y - u^0\right\|^2 + 2\sum_{i=1}^m \alpha_i[\phi(y) - \gamma_i(y)] \le \|y - u^0\|^2. \tag{4.2.5}$$

Now, assume that b) holds. Then, in view of (4.2.4), the point $\alpha(t) := (t\bar{\alpha}_1, \ldots, t\bar{\alpha}_m)$ clearly satisfies (4.2.5), and hence is feasible for (4.2.3), for every $t > 0$. Hence, for a fixed $x^* \in X^*$, this conclusion implies that

$$\sum_{i=1}^{m}(t\bar{\alpha}_i)\phi(y) \leq \sum_{i=1}^{m}(t\bar{\alpha}_i)\gamma_i(x^*) + \frac{1}{2}\|x^* - u^0\|^2 \leq t\left(\sum_{i=1}^{m}\bar{\alpha}_i\right)\phi(x^*) + \frac{1}{2}\|x^* - u^0\|^2,$$

where the last inequality follows from the assumption that $\gamma_i \leq \phi$ and the nonnegativity of $t\bar{\alpha}_1, \ldots, t\bar{\alpha}_m$. Dividing this expression by $t(\sum_{i=1}^{m}\bar{\alpha}_i) > 0$, and letting $t \to \infty$, we then conclude that $\phi(y) - \phi(x^*) \leq 0$, and hence that $y \in X^*$. Also, the objective function of (4.2.3) at the point $\alpha(t)$ converges to infinity as $t \to \infty$, showing that b) implies a).

Now assume that a) holds. Then, there exists a sequence $\{\alpha^k = (\alpha_1^k, \ldots, \alpha_m^k) : k \geq 0\}$ of feasible solutions of problem (4.2.3) along which its objective function converges to infinity. Consider the sequence $\{\bar{\alpha}^k := \alpha^k/(e^T\alpha^k) : k \geq 0\}$, where e is the vector of all ones. Clearly, $\{\bar{\alpha}^k\}$ has an accumulation point $\bar{\alpha} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_m)$, where $0 \neq \bar{\alpha} \geq 0$. Moreover, using the fact that α^k satisfies (4.2.5) and the fact that $\gamma_i(y) \leq \phi(y)$, we easily see that (4.2.4) holds. □

4.3 An adaptive accelerated method for a class of convex functions

In this section, we discuss an instance of the GenA framework for solving convex functions that, at every point $\bar{x} \in \mathcal{X}$, can be well-approximated by another convex function with an easily computable resolvent.

More specifically, in this section, we further assume that:

C.3) *there exists $L \geq 0$ such that, for every $\bar{x} \in \mathcal{X}$, we can construct a proper closed convex function $g_{\bar{x}} : \mathcal{X} \to \bar{\mathbb{R}}$ such that the resolvent $(I + \lambda\partial\bar{g}_{\bar{x}})^{-1}(x)$ is easily computable for any $x \in \mathcal{X}$ and $\lambda > 0$, and*

$$g_{\bar{x}}(x) \leq \phi(x) \leq g_{\bar{x}}(x) + \frac{L}{2}\|x - \bar{x}\|^2, \quad \forall x \in \mathcal{X}. \tag{4.3.1}$$

Clearly, (4.3.1) is equivalent to require that $\operatorname{dom} g_{\bar{x}} = \operatorname{dom}\phi$ and that the inequality in (4.3.1) holds for every $x \in \operatorname{dom}\phi$.

Before stating the method for functions ϕ satisfying C.1–C.3, we discuss some important examples of such functions.

First example: Assume that $f : \operatorname{Dom} f \subseteq \mathcal{X} \to \mathbb{R}$ is a function which is differentiable and convex on a closed convex set $\emptyset \neq \Omega \subseteq \operatorname{Dom} f$, and its gradient ∇f is L-Lipschitz continuous on Ω, i.e.,

$$\|\nabla f(x') - \nabla f(x)\| \leq L\|x' - x\|, \quad \forall x, x' \in \Omega.$$

Assume also that $h : \mathcal{X} \to \bar{\mathbb{R}}$ is a closed proper convex function such that $\operatorname{dom} h \subseteq \Omega$. Finally, assume that $\phi : \mathcal{X} \to \bar{\mathbb{R}}$ is defined as

$$\phi(x) = \begin{cases} f(x) + h(x), & \text{if } x \in \operatorname{dom} h; \\ +\infty, & \text{otherwise.} \end{cases} \tag{4.3.2}$$

Clearly, ϕ is a proper closed convex function. Moreover, the following result shows that ϕ satisfies condition C.3.

Proposition 4.8. *For any $\bar{x} \in \mathcal{X}$, the function $g_{\bar{x}} : \mathcal{X} \to \bar{\mathbb{R}}$ defined as*

$$g_{\bar{x}}(x) = f(\bar{x}_\Omega) + \langle \nabla f(\bar{x}_\Omega), x - \bar{x}_\Omega \rangle + h(x), \quad \forall x \in \mathcal{X},$$

where $\bar{x}_\Omega := \Pi_\Omega(\bar{x})$, satisfies (4.3.1).

Proof. Let $\bar{x} \in \mathcal{X}$ be given and note that $\bar{x}_\Omega := \Pi_\Omega(\bar{x}) \in \Omega$. It is well-known that the assumptions on f imply that

$$0 \leq f(x) - f(\bar{x}_\Omega) - \langle \nabla f(\bar{x}_\Omega), x - \bar{x}_\Omega \rangle \leq \frac{L}{2}\|x - \bar{x}_\Omega\|^2 \leq \frac{L}{2}\|x - \bar{x}\|^2, \quad \forall x \in \Omega,$$

where the last inequality is due to the non-expansiveness property of Π_Ω. Hence, if $x \in \operatorname{dom} \phi$, we conclude by adding $f(\bar{x}_\Omega) + \langle \nabla f(\bar{x}_\Omega), x - \bar{x}_\Omega \rangle + h(x)$ to all terms of the above relation that $g_{\bar{x}}(x) \leq \phi(x) \leq g_{\bar{x}}(x) + (L/2)\|x - \bar{x}\|^2$. Since $\operatorname{dom} \phi = \operatorname{dom} g_{\bar{x}}$, the proposition now follows from the remark following condition C.3. □

Finally, note that the resolvents of $g_{\bar{x}}$ can be computed in terms of the resolvents of h. Hence, as long as the latter can be easily computed, the function ϕ and the family of functions $\{g_{\bar{x}} : \bar{x} \in \mathcal{X}\}$ satisfies condition C.3.

Second example: Let $\Psi : \mathbb{R}^m \to \mathbb{R}$ be a closed convex function which, as a function of

each variable, is nondecreasing. Assume also that Ψ is M-Lipschitz continuous with respect to the Euclidean norm on $\mathbb{R}^m$. Let $\phi_i : \mathcal{X} \to \mathbb{R}$ together with the family $\{g_{i,\bar{x}} : \bar{x} \in \mathcal{X}\}$ satisfy C.3 with $L = L_i$, for every $i = 1, \ldots, m$. Then, it is easy to see that the function $\phi := \Psi(\phi_1, \ldots, \phi_m)$ together with the family $\{g_{\bar{x}} := \Psi(g_{1,\bar{x}}, \ldots, g_{m,\bar{x}}) : \bar{x} \in \mathcal{X}\}$ satisfy (4.3.1) with $L = M(L_1^2 + \ldots + L_m^2)^{1/2}$. Hence, as long as the resolvent of $g_{\bar{x}} := \Psi(g_{1,\bar{x}}, \ldots, g_{m,\bar{x}})$ can be computed exactly, the function ϕ and the family $\{g_{\bar{x}} : \bar{x} \in \mathcal{X}\}$ will satisfy C.3.

We will now present an instance of the GenA framework for functions ϕ satisfying conditions C.1–C.3. The actual proof that the method below is an instance of the GenA framework will given in Proposition 4.10.

Algorithm 4.1 Adaptive accelerated (AA) method for (1.2.1)

0) Let $y^0, u^0 \in \mathcal{X}$, $\tau \in (0,1]$, $A_0 \geq 0$, and an affine function $\Gamma_0 : \mathcal{X} \to \mathbb{R}$ such that

$$A_0\Gamma_0 \leq A_0\phi, \quad A_0\phi(y^0) \leq \inf_{x\in\mathcal{X}} \left\{ A_0\Gamma_0(x) + \frac{1}{2}\|x - u^0\|^2 \right\},$$

be given, and set $\lambda := \tau/L$, $\Gamma_0^0 = \Gamma_0(u^0)$, $V_0 = \nabla\Gamma_0$ and $k = 1$;

1) set $x^k = u^0 - A_k V_k$,

$$a_{k+1} = \frac{\lambda + \sqrt{\lambda^2 + 4\lambda A_k}}{2}, \quad \tilde{y}^{k+1} = \arg\min_{\tilde{x}\in\mathcal{X}} \left\{ \lambda g_{\tilde{x}^k}(\tilde{x}) + \frac{1}{2}\|\tilde{x} - \tilde{x}^k\|^2 \right\}, \tag{4.3.3}$$

where $g_{\tilde{x}^k}$ is as in condition C.3 above and

$$\tilde{x}^k = \frac{A_k}{A_k + a_{k+1}}\, y^k + \frac{a_{k+1}}{A_k + a_{k+1}}\, x^k;$$

2) set

$$v^{k+1} := \frac{1}{\lambda}(\tilde{x}^k - \tilde{y}^{k+1}), \qquad \gamma_{k+1}^0 = g_{\tilde{x}^k}(\tilde{y}^{k+1}) + \langle u^0 - \tilde{y}^{k+1}, v^{k+1}\rangle; \tag{4.3.4}$$

3) choose $y^{k+1} \in \mathcal{X}$ such that $\phi(y^{k+1}) \leq \phi(\tilde{y}^{k+1})$ and the pair $(A'_k, a'_{k+1}) \in \mathbb{R}^2$ as an optimal solution of

$$\begin{array}{ll} \max & A' + a' \\ \text{s.t.} & \frac{1}{2}\|A'V_k + a'v^{k+1}\|^2 + A'[\phi(y^{k+1}) - \Gamma_k^0] + a'[\phi(y^{k+1}) - \gamma_{k+1}^0] \leq 0, \\ & A', a' \geq 0. \end{array} \tag{4.3.5}$$

with the safeguard that $A_0 = A'_0$ when $k = 0$; if the above problem is unbounded, then **stop** by declaring y^{k+1} to be an optimal solution of (1.2.1);

4) set $A_{k+1} = A'_k + a'_{k+1}$,

$$\Gamma_{k+1}^0 = \frac{A'_k}{A'_k + a'_{k+1}}\Gamma_k^0 + \frac{a'_{k+1}}{A'_k + a'_{k+1}}\gamma_{k+1}^0, \quad V_{k+1} = \frac{A'_k}{A'_k + a'_{k+1}}V_k + \frac{a'_{k+1}}{A'_k + a'_{k+1}}v^{k+1}, \tag{4.3.6}$$

$k \leftarrow k + 1$, and go to step 1.

We now make a remark about the AA method for the case where ϕ and $g_{\tilde{x}}$ are as in

the first example at the beginning of this section. For this case, it can be shown that when $\Omega = \mathcal{X}$, and hence ∇f is defined and is L-Lipschitz continuous on the whole $\mathcal{X}$, the AA method reduces to a well-known variant of Nesterov's method, namely FISTA [1] (see also Algorithm 2 of [65]).

Before establishing the convergence properties of the AA method, we state and prove two technical results.

Lemma 4.9. *Let $\lambda > 0$, $\xi \in \mathcal{X}$ and a proper closed convex function $f : \mathcal{X} \to (-\infty, \infty]$ be given and denote the optimal solution and optimal value of*

$$\min_{x \in \mathcal{X}} \left\{ \lambda f(x) + \frac{1}{2}\|x - \xi\|^2 \right\}. \tag{4.3.7}$$

by $\bar{x}$ and $\bar{f}$, respectively. Then, the affine function $\mathcal{A} : \mathcal{X} \to \mathbb{R}$ defined as

$$\mathcal{A}(x) = f(\bar{x}) + \frac{1}{\lambda}\langle \xi - \bar{x}, x - \bar{x} \rangle, \quad \forall x \in \mathcal{X},$$

has the property that $\mathcal{A} \leq f$ and the optimal solution and optimal value of

$$\min_{x \in \mathcal{X}} \left\{ \lambda \mathcal{A}(x) + \frac{1}{2}\|x - \xi\|^2 \right\} \tag{4.3.8}$$

is $\bar{x}$ and $\bar{f}$, respectively.

Proof. The optimality condition for (4.3.7) says that $0 \in \lambda \partial f(\bar{x}) + \bar{x} - \xi$, or equivalently

$$\frac{1}{\lambda}(\xi - \bar{x}) \in \partial f(\bar{x}),$$

and hence that $\mathcal{A} \leq f$, by the definition of $\mathcal{A}$ and the subgradient of a function. Moreover, $\bar{x}$ clearly satisfies the optimality condition of (4.3.8). Hence, the last claim of the proposition follows. □

The following result shows that the AA method is a special instance of the GenA framework.

Proposition 4.10. *Consider the AA method and, for every $k \geq 0$, define*

$$\lambda_{k+1} = \lambda := \frac{\tau}{L}, \qquad \gamma_{k+1}(x) = g_{\tilde{x}^k}(\tilde{y}^{k+1}) + \langle x - \tilde{y}^{k+1}, v^{k+1} \rangle, \tag{4.3.9}$$

where v^{k+1} is defined in (4.3.4). Also, define the sequence of functions $\{\Gamma_k\}$ recursively as in (4.1.8). Then, for every $k \geq 0$, (4.1.3), (4.1.5) and (4.1.7) hold. In particular, the AA method is a special case of the GenA framework.

Proof. Using the first relation in (4.3.3) and the definition of λ_{k+1} in (4.3.9), it follows that (4.1.5) trivially holds. Clearly, in view of (4.3.9) and (4.3.4), we have $v^{k+1} = \nabla\gamma_{k+1}$ and $\gamma_{k+1}^0 = \gamma_{k+1}(u^0)$ for every $k \geq 0$. Hence, the initialization in step 0 of the AA method, and the recursive definition of $\{\Gamma_k\}$ in (4.1.8) together with (4.3.6), imply that $\Gamma_k^0 = \Gamma_k(u^0)$ and $V_k = \nabla\Gamma_k$ for all $k \geq 0$. The above two conclusions then imply that the convex quadratic constraint in (4.3.5) is equivalent to (4.2.1), which in turn is equivalent to (4.1.7), in view of Proposition 4.6. We will now show that (4.1.3) holds for every $k \geq 0$. Indeed, in view of the second relation in (4.3.3), the definition of λ_{k+1} in (4.3.9), and condition C.3 with $\bar{x} = \tilde{x}^k$ and $x = \tilde{y}^{k+1}$, we have

$$\begin{aligned}\min_{\tilde{x}\in\mathcal{X}}\left\{\lambda_{k+1}g_{\tilde{x}^k}(\tilde{x}) + \frac{1}{2}\|\tilde{x}-\tilde{x}^k\|^2\right\} &= \lambda_{k+1}g_{\tilde{x}^k}(\tilde{y}^{k+1}) + \frac{1}{2}\|\tilde{y}^{k+1}-\tilde{x}^k\|^2 \\ &= \lambda_{k+1}\left[g_{\tilde{x}^k}(\tilde{y}^{k+1}) + \frac{L}{2}\|\tilde{y}^{k+1}-\tilde{x}^k\|^2\right] + \frac{1}{2}\left(1-\lambda_{k+1}L\right)\|\tilde{y}^{k+1}-\tilde{x}^k\|^2 \\ &\geq \lambda\phi(\tilde{y}^{k+1}) + \frac{1}{2}(1-\tau)\|\tilde{y}^{k+1}-\tilde{x}^k\|^2.\end{aligned}$$

Now, using the definition of γ_{k+1} in (4.3.9), Lemma 4.9 with $f = g_{\tilde{x}^k}$ and $\xi = \tilde{x}^k$, and noting that $\bar{x} = \tilde{y}^{k+1}$ in this case, we have

$$\min_{\tilde{x}\in\mathcal{X}}\left\{\lambda_{k+1}g_{\tilde{x}^k}(\tilde{x}) + \frac{1}{2}\|\tilde{x}-\tilde{x}^k\|^2\right\} = \min_{\tilde{x}\in\mathcal{X}}\left\{\lambda_{k+1}\gamma_{k+1}(\tilde{x}) + \frac{1}{2}\|\tilde{x}-\tilde{x}^k\|^2\right\}.$$

Combining the above two relations, we then conclude that (4.1.3) holds. □

The following convergence result follows as an immediate consequence of Theorem 4.5, Proposition 4.10, and the fact that $\lambda_k = \tau L^{-1}$ for every $k \geq 1$.

Theorem 4.11. *Let x^* be the projection of u^0 onto X^* and d_0 be the distance of u^0 to X^*. Then, for every $k \geq 1$,*

$$\phi(y^k) - \phi^* \leq \frac{d_0^2}{2A_k} \leq \frac{2Ld_0^2}{\tau k^2}, \qquad \|x^k - x^*\| \leq d_0.$$

4.4 Numerical results

In this section, we describe two restarting variants of the AA method and report numerical results comparing them to the following variants of Nesterov's method:

i) FISTA (fast iterative shrinkage-thresholding algorithm) of [1] (see also Algorithm 2 of [65]);

ii) FISTA-R: restarting variant of FISTA;

iii) GKR-2: Algorithm 2 of In [20];

iii) GKR-3: Algorithm 3 of [20].

More specifically, we compare the performance of these methods using three classes of conic quadratic programming instances, namely:

a) random convex quadratic programs (CQPs) (see Subsection 4.4.1);

b) semidefinite least squares (SDLSs) (see Subsection 4.4.2); and

c) random nonnegative least squares (NNLSs) (see Subsection 4.4.3).

We observe that we have implemented our own code for FISTA-R. Although we have recently learned that a similar variant was implemented in [51], we have not had the opportunity to include their code in our computational benchmark. Nevertheless, we believe that our implementation should be very similar to the method in [51], and hence should reflect the actual performance of the latter algorithm.

We stop all methods whenever an iterate y^k is found such that

$$\left\| y^k - (I + \partial h)^{-1} \left(y^k - \frac{1}{L} \nabla g(y^k) \right) \right\| \leq 10^{-6} \tag{4.4.1}$$

or when 2000 iterations have been performed. Note that for the case where h is an indicator function of a closed convex set, the left hand side of (4.4.1) is exactly the norm of the projected gradient with stepsize $1/L$.

We now briefly describe the two restarting versions of the AA method. In the first version, if the function value at the end of the kth iteration increases, i.e., $\phi(y^k) < \phi(y^{k+1})$, then the method is restarted at step 0 with $u^0 = y^k$, $y^0 = y^k$, $A_0 = 0$, $\Gamma_0^0 = 0$ and $V_0 = 0$. We observe that we have implemented FISTA-R by incorporating this restarting scheme to FISTA.

In contrast to the first restarting version above, the second one allows some iterations to increase the function value, and hence is more conservative than the first one. More specifically, we restart this variant at the kth iteration whenever $\phi(y^k) < \phi(y^{k+1})$ and no more than $\lceil \log_2(k - l_k) \rceil$ restarts have been performed so far, where l_k is the iteration where the last restart was performed.

We will refer to the first and second restarting variants of the AA method as AA-R1 and AA-R2, respectively.

The codes for all the benchmarked variants tested are written in Matlab. All the computational results were obtained on a single core of a server with 2 Xeon X5520 processors at 2.27GHz and 48GB RAM.

We now make some general remarks about how the results are reported on the tables given below. Tables 4.4.1, 4.4.3 and 4.4.5 report the times and Tables 4.4.2, 4.4.4 and 4.4.6 report the number of iterations for all instances of the three problem classes. Each problem class is associated with two tables, one reporting the times and the other one the number of iterations required by each benchmarked method to solve all instances of the class. We display the time or number of iterations that a variant takes on an instance in red, and also with an asterisk (*), whenever it can not solve the instance to the required accuracy. In such a case, the accuracy obtained at the last iteration of the variant is also displayed in parentheses.

Figures 4.4.1, 4.4.3 and 4.4.5 plot the time performance profiles (see [15]), and Figures 4.4.2, 4.4.4 and 4.4.6 plot the iteration performance profiles for each of the three problem classes. We recall the following definition of a performance profile. For a given instance, a method A is said to be at most x times slower than method B, if the time taken (resp. number of iterations performed) by method A is at most x times the time taken (resp. number of iterations performed) by method B. A point (x, y) is in the performance profile curve of a method if it can solve exactly $(100y)\%$ of all the tested instances x times slower than any other competing method.

4.4.1 Numerical results for random CQPs

This subsection compares the performance of our methods AA-R1 and AA-R2 with the variants of Nesterov's method listed at the beginning of this section on a class of randomly generated sparseCQP instances. These instances were also used to report the performance of GKR-2 and GKR-3 in [20].

Given $Q \in S^n_+$, $b \in \mathbb{R}^n$, $l \in \{\mathbb{R} \cup \{-\infty\}\}^n$ and $u \in \{\mathbb{R} \cup \{\infty\}\}^n$ such that $l \leq u$, the box constrained convex quadratic programming problem is defined as

$$\min_{x \in \mathbb{R}^n} \left\{ x^T Q x + b^T x \ : \ l \leq x \leq u \right\}. \tag{4.4.2}$$

Letting f and h be defined as

$$f(x) = x^T Q x + b^T x, \qquad h(x) = \delta_B(x), \qquad \forall x \in \mathbb{R}^n,$$

where $B = \{x \in \mathbb{R}^n : l \leq x \leq u\}$, we can easily see that (4.4.2) is a special case of (1.2.1) with $\mathcal{X} = \mathbb{R}^n$ and ϕ given by (4.3.2).

Figures 4.4.1 and 4.4.2 plot time and iteration performance profiles of all variants of Nesterov's method for solving this collection of random sparseCQP instances, respectively. Tables 4.4.1 and 4.4.2 report the time and number of iterations taken by each method, respectively.

4.4.2 Numerical results for SDLSs

This subsection compares the performance of our methods AA-R1 and AA-R2 with the variants of Nesterov's method listed at the beginning of this section on a class of SDLS instances.

Let $\mathcal{S}^n$ be the set of all $n \times n$ symmetric matrices and $\mathcal{S}^n_+$ be the cone of $n \times n$ symmetric positive semidefinite matrices. Given a linear map $\mathcal{A} \in \mathcal{S}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$, the SDP feasibility problem consists of finding x such that

$$\mathcal{A}x = b, \qquad x \in \mathcal{S}^n_+.$$

We can solve the above problem by considering the SDLS reformulation

$$\min_{x \in \mathcal{S}^n} \left\{ \frac{1}{2} \left\| \mathcal{A}x - b \right\|^2 \ : \ x \in \mathcal{S}^n_+ \right\}. \tag{4.4.3}$$

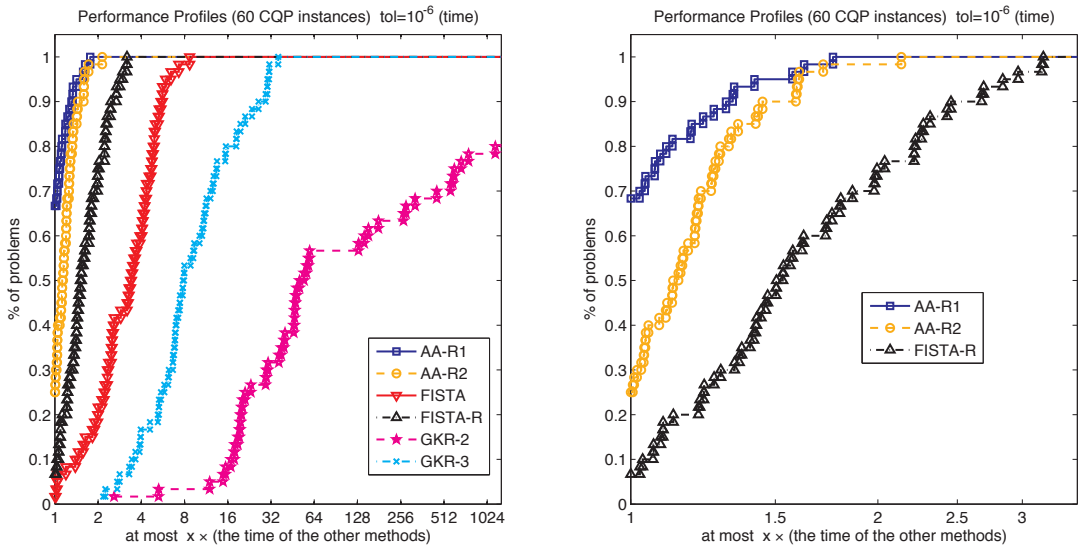

Figure 4.4.1: Time performance profiles for solving CQP instances with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

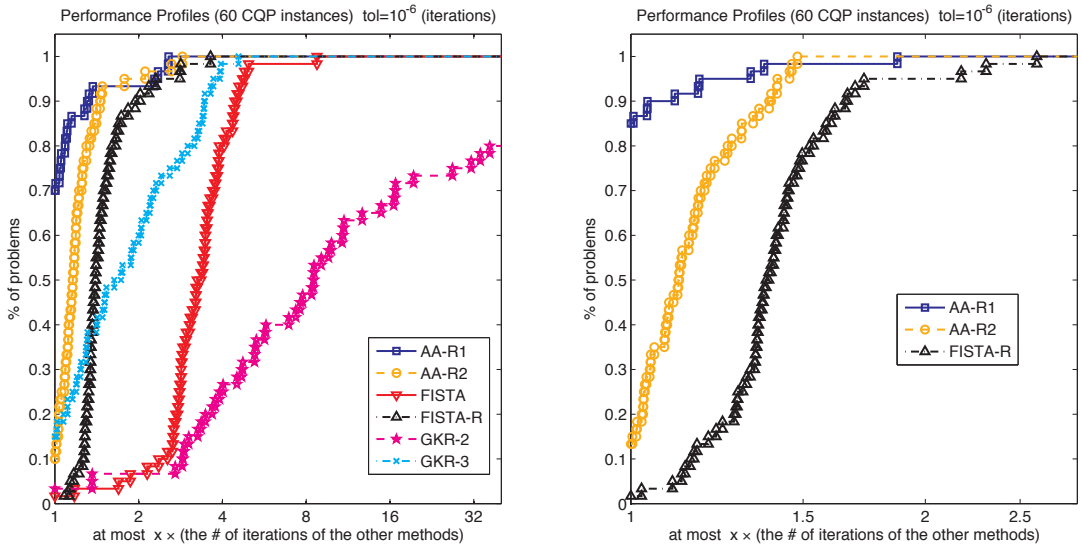

Figure 4.4.2: Iteration performance profiles for solving CQP instances with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

Table 4.4.1: Time comparison of the methods on CQP instances.

Problem		Time in seconds (accuracy less that 10^{-6})					
Instance	n	**AA-R1**	**AA-R2**	**FISTA**	**FISTA-R**	**GKR-2**	**GKR-3**
QP_n5_p2	5	0.1	0.1	0.1	0.1	9.9	0.5
QP_n5_p3	5	0.1	0.1	0.1	0.1	0.4	0.3
QP_n5_p4	5	0.1	0.1	0.3	0.2	16.3	2.4
QP_n5_p5	5	0.2	0.1	0.3	0.2	7.5	0.9
QP_n10_p2	10	0.1	0.1	0.1	0.1	4.7	0.5
QP_n10_p3	10	0.1	0.1	0.2	0.1	0.2	0.5
QP_n10_p4	10	0.2	0.2	0.3	0.2	99.5*(1.2 -6)	1.3
QP_n10_p5	10	0.5	0.6	0.4	0.7	20.1	7.9
QP_n20_p2	20	0.1	0.1	0.1	0.1	24.8	0.3
QP_n20_p3	20	0.1	0.1	0.1	0.1	1.8	0.4
QP_n20_p4	20	0.2	0.1	0.3	0.2	80.1	4.2
QP_n20_p5	20	0.4	0.5	0.4	0.6	76.6	12.9
QP_n50_p2	50	0.1	0.1	0.1	0.1	44.8	0.3
QP_n50_p3	50	0.1	0.2	0.4	0.2	29.5	1.1
QP_n50_p4	50	0.2	0.2	0.2	0.1	5.8	3
QP_n50_p5	50	0.2	0.2	0.4	0.3	27.4*(1.3 -6)	6.1
QP_n100_p2	100	0.1	0.1	0.2	0.1	46.2	0.5
QP_n100_p3	100	0.2	0.2	0.4	0.2	7.5	1.7
QP_n100_p4	100	0.2	0.2	0.8	0.2	121.5	2.5
QP_n100_p5	100	0.3	0.3	0.7	0.3	138.2*(2.1 -6)	8.5
QP_n200_p2	200	0.1	0.1	0.2	0.1	28.2*(1.6 -6)	0.8
QP_n200_p3	200	0.2	0.2	0.5	0.2	40.3	1.1
QP_n200_p4	200	0.2	0.2	0.7	0.3	3.3	2.5
QP_n200_p5	200	0.3	0.4	0.7	0.4	47.9	7.4
QP_n500_p2	500	0.1	0.2	0.4	0.1	145.4	0.9
QP_n500_p3	500	0.3	0.3	1.2	0.3	10.8	2.4
QP_n500_p4	500	0.3	0.4	1.4	0.6	14.7	4.3
QP_n500_p5	500	0.4	0.5	2.1	1	23.7	12.8
QP_n700_p2	700	0.2	0.2	0.6	0.2	119.9	1.1
QP_n700_p3	700	0.3	0.4	1.9	0.6	115.9	3.1
QP_n700_p4	700	0.4	0.4	3.1	1.1	18.3	7.6
QP_n700_p5	700	1.4	1.1	5.4	1.1	19.1	19.6
QP_n900_p2	900	0.3	0.3	0.8	0.7	120.0*(2.2 -6)	1.7
QP_n900_p3	900	0.6	0.6	2.9	0.9	28.5	4
QP_n900_p4	900	0.9	1	3.8	1.4	26.4	9.7
QP_n900_p5	900	1.6	1.9	6.9	2.8	31.3	24.9
QP_n1K_p2	1000	0.4	0.4	1.3	0.6	84.3*(4.0 -6)	4.3
QP_n1K_p3	1000	0.7	0.7	3.1	1.2	31.5	5
QP_n1K_p4	1000	1.1	1.5	5.6	1.6	20	8.4
QP_n1K_p5	1000	2.5	1.8	8.7	3	63.8	63.1
QP_n2K_p2	2000	1	1.2	4.3	2.8	112.0*(3.1 -6)	3.9
QP_n2K_p3	2000	1.8	2.9	10.3	3.7	88.6	11.2
QP_n2K_p4	2000	3	3.1	17.8	6	120.8	23.4
QP_n2K_p5	2000	9.5	9.6	23.6	10.4	114	68.9
QP_n5K_p2	5000	8.7	7.1	17.9	14.1	863.0*(3.2 -6)	20.2
QP_n5K_p3	5000	20.3	19	48.5	29.1	400.7	42.6
QP_n5K_p4	5000	23.6	37.8	129.1	46.7	471.1	161.4
QP_n5K_p5	5000	47.8	40.3	146.6	91	597.6	513.9
QP_n7K_p2	7000	18.9	25.1	70.5	46	1544.8*(4.4 -6)	65.2
QP_n7K_p3	7000	31.4	37.8	128.1	45.9	923.9	85.5
QP_n7K_p4	7000	25.9	29.2	167	63.7	1082.3*(1.2 -6)	264.2
QP_n7K_p5	7000	64.4	102.6	338.5	115.8	1307.1	734
QP_n9K_p2	9000	25.4	40.3	58.9	68.3	3071.3*(5.5 -6)	100.6
QP_n9K_p3	9000	64.1	54.2	219	120	2046.5	287.3
QP_n9K_p4	9000	95	86	483.1	129.2	1992	669.5
QP_n9K_p5	9000	128.8	163.3	530.7	292.7	2165.3	1203.9
QP_n10K_p2	10000	107.9	81.1	84.2	240.4	3519.7*(5.5 -6)	176.1
QP_n10K_p3	10000	84	64.3	251.8	113	1911.6	250.1
QP_n10K_p4	10000	173.8	106.9	365.4	236.9	2089.7	733.5
QP_n10K_p5	10000	141.8	202.6	688.5	451.3	2190.9	1784

Table 4.4.2: Number of iterations comparison of the methods on CQP instances.

Problem Instance	n	# of iterations (accuracy less that 10^{-6}) AA-R1	AA-R2	FISTA	FISTA-R	GKR-2	GKR-3
QP_n5_p2	5	26	25	59	35	913	48
QP_n5_p3	5	41	38	53	35	18	20
QP_n5_p4	5	89	92	314	104	1491	195
QP_n5_p5	5	176	94	215	152	397	75
QP_n10_p2	10	39	44	94	40	315	37
QP_n10_p3	10	41	46	140	58	16	44
QP_n10_p4	10	130	95	254	146	2000*(1.2 -6)	114
QP_n10_p5	10	462	483	425	536	579	907
QP_n20_p2	20	23	32	65	33	483	18
QP_n20_p3	20	47	40	61	46	49	36
QP_n20_p4	20	118	89	289	114	1132	407
QP_n20_p5	20	374	360	423	459	1041	1422
QP_n50_p2	50	27	36	58	33	525	29
QP_n50_p3	50	64	93	289	90	334	95
QP_n50_p4	50	126	132	353	164	873	416
QP_n50_p5	50	183	165	308	247	2000*(1.3 -6)	398
QP_n100_p2	100	37	48	124	49	618	46
QP_n100_p3	100	88	93	338	118	948	176
QP_n100_p4	100	116	130	576	186	1726	256
QP_n100_p5	100	232	235	659	321	2000*(2.1 -6)	861
QP_n200_p2	200	41	52	140	61	2000*(1.6 -6)	62
QP_n200_p3	200	69	80	326	96	525	104
QP_n200_p4	200	134	139	584	183	391	219
QP_n200_p5	200	232	259	645	298	628	803
QP_n500_p2	500	57	74	151	77	1926	58
QP_n500_p3	500	110	134	406	125	939	155
QP_n500_p4	500	164	163	710	241	1384	307
QP_n500_p5	500	293	294	1017	377	1374	1063
QP_n700_p2	700	51	72	163	111	1598	63
QP_n700_p3	700	95	134	421	130	1567	178
QP_n700_p4	700	169	189	686	238	801	346
QP_n700_p5	700	336	346	929	462	1168	1008
QP_n900_p2	900	52	72	179	120	2000*(2.2 -6)	62
QP_n900_p3	900	105	131	398	143	1148	137
QP_n900_p4	900	155	168	641	225	889	350
QP_n900_p5	900	320	372	998	431	1443	1086
QP_n1K_p2	1000	58	68	197	83	2000*(4.0 -6)	76
QP_n1K_p3	1000	111	125	433	169	1206	170
QP_n1K_p4	1000	164	192	721	258	857	379
QP_n1K_p5	1000	329	377	1274	432	996	1289
QP_n2K_p2	2000	63	76	187	85	2000*(3.1 -6)	70
QP_n2K_p3	2000	133	159	428	179	1113	172
QP_n2K_p4	2000	160	185	769	277	1540	414
QP_n2K_p5	2000	313	341	1102	438	1021	1076
QP_n5K_p2	5000	83	87	215	103	2000*(3.2 -6)	79
QP_n5K_p3	5000	194	199	453	263	1240	169
QP_n5K_p4	5000	290	299	825	388	1066	412
QP_n5K_p5	5000	356	447	1265	522	1391	1161
QP_n7K_p2	7000	85	94	219	102	2000*(4.4 -6)	77
QP_n7K_p3	7000	136	138	473	195	1341	176
QP_n7K_p4	7000	190	226	871	275	2000*(1.2 -6)	388
QP_n7K_p5	7000	369	539	1385	611	1392	1264
QP_n9K_p2	9000	85	93	225	221	2000*(5.5 -6)	78
QP_n9K_p3	9000	152	225	463	204	1378	176
QP_n9K_p4	9000	299	343	916	337	1203	523
QP_n9K_p5	9000	397	537	1379	673	1333	1259
QP_n10K_p2	10000	204	210	225	225	2000*(5.5 -6)	80
QP_n10K_p3	10000	183	198	487	264	1422	186
QP_n10K_p4	10000	236	258	918	393	1338	408
QP_n10K_p5	10000	459	541	1473	727	1318	1316

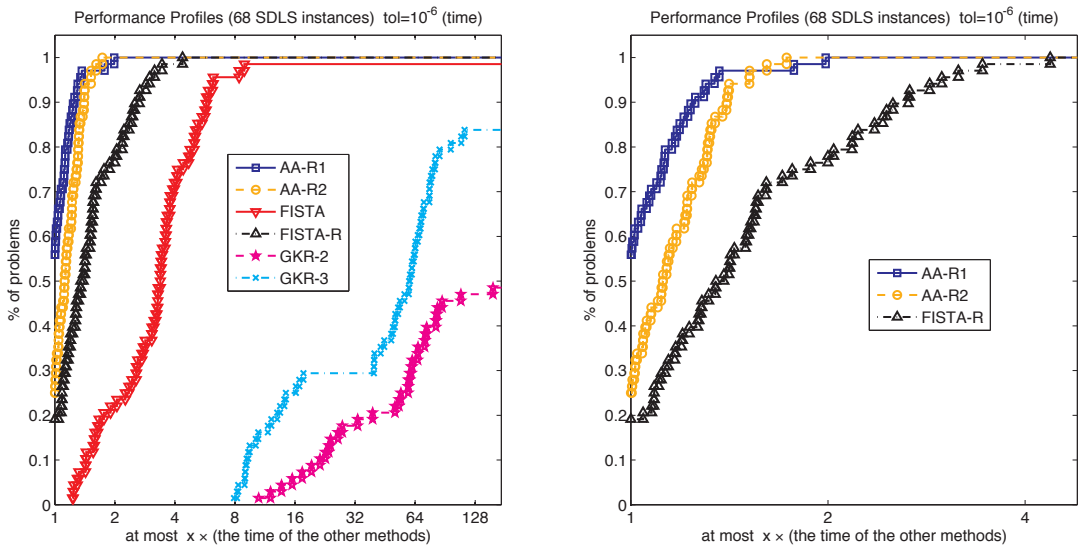

Figure 4.4.3: Time performance profiles for solving SDLS instances with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

Letting f and h be defined as

$$f(x) = \frac{1}{2}\|\mathcal{A}x - b\|^2, \qquad h(x) = \delta_{S^n_+}(x), \qquad \forall x \in \mathcal{S}^n,$$

where $\|\cdot\|$ denotes the Euclidean norm, we can easily see that (4.4.3) is a special case of (1.2.1) with $\mathcal{X} = \mathcal{S}^n$ and ϕ given by (4.3.2).

The SDLS instances included in this comparison are obtained via the above construction from the feasibility sets (after bringing them into standard form) of four classes of SDPs , namely: i) randomly generated SDPs as in [52]; ii) SDP relaxations of frequency assignment problems as in (2.2.25); iii) SDP relaxations of binary integer quadratic problems as in (2.2.26); and iv) SDP relaxations of quadratic assignment problems as in (2.2.27).

Figures 4.4.3 and 4.4.4 plot time and iteration performance profiles of all variants of Nesterov's method for solving this collection of SDLS instances, respectively. Tables 4.4.3 and 4.4.4 report the time and number of iterations taken by each method, respectively.

4.4.3 Numerical results for NNLSs

This subsection compares the performance of our methods AA-R1 and AA-R2 with the variants of Nesterov's method listed at the beginning of this section on a class of NNLS

Table 4.4.3: Time comparison of the methods on SDLS instances.

Problem		Time in seconds (accuracy less that 10^{-6})					
Instance	n	**AA-R1**	**AA-R2**	**FISTA**	**FISTA-R**	**GKR-2**	**GKR-3**
BIQ_n21_m252	21	1	0.7	2.5	1	402.9*(1.4 -6)	46.8
BIQ_n31_m527	31	1	0.8	4	1.2	248.6*(1.4 -6)	47.6
BIQ_n41_m902	41	1	0.8	2.7	1	481.6*(1.4 -6)	47.8
BIQ_n51_m1377	51	1	1	2.8	1.1	285.3*(1.4 -6)	44.4
BIQ_n61_m1952	61	0.9	1.1	3.7	1.2	537.2*(1.4 -6)	42.1
BIQ_n71_m2627	71	0.8	1.2	3.1	1.2	548.0*(1.4 -6)	52.8
BIQ_n81_m3402	81	1	1.1	4.2	1.5	266.3*(1.4 -6)	40.4
BIQ_n91_m4277	91	1	1.2	3.2	1.5	279.0*(1.4 -6)	48.7
BIQ_n101_m5252	101	0.9	1.5	3.2	1.4	360.4*(1.4 -6)	56.7
BIQ_n121_m7502	121	1.1	1.1	6.3	1.7	695.4*(1.4 -6)	44.6
BIQ_n151_m11627	151	1.3	1.5	4.5	1.8	939.9*(1.4 -6)	64.5
BIQ_n201_m20502	201	2.7	2.1	6.5	3.1	707.5*(1.4 -6)	157.9
BIQ_n251_m31877	251	2.7	3.1	15.9	4	1073.8*(1.4 -6)	144.8
BIQ_n501_m126252	501	16.3	13.9	52.3	17.7	6386.8*(1.4 -6)	706
FAP_n52_m1378	52	0.4	0.5	0.5	0.4	5.6	3.7
FAP_n61_m1866	61	0.4	0.5	0.5	0.5	6.5	3.5
FAP_n65_m2145	65	0.5	0.6	0.6	0.5	11.2	3.9
FAP_n81_m3321	81	0.5	0.5	0.7	0.6	5.4	4.7
FAP_n84_m3570	84	0.5	0.6	0.6	0.5	5.9	4.4
FAP_n93_m4371	93	0.5	0.7	0.8	0.7	15.3	4.3
FAP_n98_m4851	98	0.5	0.7	0.8	0.6	8	4.4
FAP_n120_m7260	120	0.7	0.9	1	0.8	17	5.6
FAP_n174_m15225	174	0.9	1.2	2.3	1.1	21.1	8.7
FAP_n183_m14479	183	1	1.4	1.9	1.5	38.9	10.4
FAP_n252_m24292	252	2.1	2.6	3.2	2.3	39.8	20.9
FAP_n369_m26462	369	3.7	4.3	6.1	4	78.6	44.8
FAP_n2118_m322924	2118	354.3	466.9	1169	550.5	9788	5722.2
FAP_n4110_m1154467	4110	2518.5	3039.2	5151.6	3187	66187	29440.6
QAP_n144_m10672	144	6.8	7.5	44.2*(1.1 -6)	5.6	807.3*(2.5 -6)	544.1
QAP_n196_m19619	196	12	10.7	24.1	11.1	1150.6*(3.3 -6)	1299.8*(2.3 -6)
QAP_n225_m25783	225	25.7	13	18.6	14	689.2	186.8
QAP_n256_m33302	256	21.5	21.7	24.8	15.7	2265.5*(1.7 -6)	3289.2*(1.2 -6)
QAP_n289_m42362	289	28	25.9	53.7	22.9	2496.2*(4.0 -6)	1727.5
QAP_n324_m53161	324	34	36.1	98.5	27.6	3220.6*(1.9 -6)	3124.4
QAP_n400_m80828	400	58.5	53.4	440.8	52.6	4832.9*(2.4 -6)	5571.4*(1.3 -6)
QAP_n441_m98152	441	68.2	76.3	339.9	59	4207.4	12851.7*(1.8 -6)
QAP_n484_m118127	484	95.3	113.9	434.4	87.1	5387	1133.6
QAP_n625_m196598	625	282.9	210.7	1420.3	159.4	14179.8*(2.8 -6)	2809
QAP_n676_m229877	676	212.3	263.3	543.1	247.6	16715.0*(2.7 -6)	3566.5
QAP_n729_m267217	729	243	270.5	376.6	236.9	20852.3*(1.8 -6)	20097.9
QAP_n784_m308936	784	333	328.7	1384.4	295.4	24197.1*(2.0 -6)	23049.4
QAP_n900_m406843	900	480.7	516.7	1678.3	464.1	23520.9	6558.9
QAP_n1225_m752813	1225	1258	1170.2	2873.9	1121	99556.0*(1.3 -6)	15090.9
QAP_n1600_m1283258	1600	2512.7	2811.8	3610.1	3014.3	207778.0*(2.3 -6)	171500.0*(1.1 -6)
RAND_n300_m10K_p4	300	13	12.3	31.9	16.1	810.3	490.3
RAND_n300_m20K_p3	300	43.5	46.7	139.6	68.1	2918.1*(1.1 -6)	2987.4
RAND_n300_m25K_p3	300	41.6	48.9	157.7	143.3	2981.1*(1.5 -6)	2682.2*(1.0 -6)
RAND_n400_m15K_p4	400	15.7	27.2	96.8	27.7	851.8	794.4
RAND_n400_m30K_p3	400	82.3	94.4	362.2	168	4951.4	8962.2
RAND_n400_m40K_p3	400	80.1	84.7	273.2	178	9705.8*(1.5 -6)	5465.2*(1.3 -6)
RAND_n500_m20K_p4	500	38.4	58.3	117.9	47.2	2274.4	2292.9
RAND_n500_m30K_p3	500	100.1	104.1	306.4	141.3	6594.5	5263
RAND_n500_m40K_p3	500	113.7	119.6	576.4	247.7	8313	7979.3
RAND_n500_m50K_p3	500	192.9	148.3	411.9	647.7	10854.6*(1.1 -6)	10096.9*(1.3 -6)
RAND_n600_m20K_p4	600	36.8	37.6	120.8	73.6	2175.8	2349.3
RAND_n600_m40K_p3	600	233.1	199.5	816.2	321.3	12218	12037.4
RAND_n600_m50K_p3	600	168.9	166.9	571.8	445	13565.8	11012.5
RAND_n600_m60K_p3	600	218.8	265	792.8	411.6	17428.4*(1.6 -6)	15356.9*(1.2 -6)
RAND_n700_m50K_p3	700	236	239.5	1124.4	402.1	20615.5	17822.2
RAND_n700_m70K_p3	700	258.8	241	1383	592.2	19660.1	16415.1
RAND_n700_m90K_p3	700	352.7	317.7	1075.1	948.2	21967.6*(1.3 -6)	22553.8*(1.1 -6)
RAND_n800_m100K_p3	800	422.3	641.1	1514.7	1024	30362.9	25120.1
RAND_n800_m110K_p3	800	403	405.1	2206.4	1152.5	30609.5*(1.2 -6)	30426.3*(1.1 -6)
RAND_n800_m70K_p3	800	289.9	310.8	1803	728.4	45568.8	21851.4
RAND_n900_m100K_p3	900	589.3	587.5	1666.9	1278.9	48650.9	39438.5
RAND_n900_m140K_p3	900	552.5	577	2767.4	1465.5	46800.9*(1.3 -6)	39116.8
RAND_n1K_m100K_p3	1000	568	532.4	2074.3	1683.9	57775.2	43090
RAND_n1K_m150K_p3	1000	768.4	777.7	2810.2	1809.5	63396.8*(1.2 -6)	61719.5

Table 4.4.4: Number of iterations comparison of the methods on SDLS instances.

Problem Instance	n	# of iterations (accuracy less that 10^{-6}) AA-R1	AA-R2	FISTA	FISTA-R	GKR-2	GKR-3
BIQ_n21_m252	21	79	67	251	90	2000*(1.4 -6)	285
BIQ_n31_m527	31	79	67	251	90	2000*(1.4 -6)	285
BIQ_n41_m902	41	79	67	251	90	2000*(1.4 -6)	285
BIQ_n51_m1377	51	79	67	251	90	2000*(1.4 -6)	285
BIQ_n61_m1952	61	68	73	251	90	2000*(1.4 -6)	285
BIQ_n71_m2627	71	65	68	251	90	2000*(1.4 -6)	285
BIQ_n81_m3402	81	68	73	251	90	2000*(1.4 -6)	285
BIQ_n91_m4277	91	65	68	251	90	2000*(1.4 -6)	285
BIQ_n101_m5252	101	67	72	251	90	2000*(1.4 -6)	285
BIQ_n121_m7502	121	65	67	251	90	2000*(1.4 -6)	285
BIQ_n151_m11627	151	68	73	251	90	2000*(1.4 -6)	285
BIQ_n201_m20502	201	79	67	251	90	2000*(1.4 -6)	285
BIQ_n251_m31877	251	64	66	251	90	2000*(1.4 -6)	285
BIQ_n501_m126252	501	65	67	251	90	2000*(1.4 -6)	285
FAP_n52_m1378	52	25	27	37	26	29	18
FAP_n61_m1866	61	21	25	38	26	32	18
FAP_n65_m2145	65	25	27	38	26	46	18
FAP_n81_m3321	81	23	23	37	26	23	19
FAP_n84_m3570	84	21	26	38	26	22	19
FAP_n93_m4371	93	20	29	39	26	43	18
FAP_n98_m4851	98	21	27	43	26	36	18
FAP_n120_m7260	120	24	28	43	26	38	18
FAP_n174_m15225	174	23	29	46	26	43	18
FAP_n183_m14479	183	21	26	44	26	41	19
FAP_n252_m24292	252	26	31	45	26	35	20
FAP_n369_m26462	369	24	27	47	26	40	20
FAP_n2118_m322924	2118	25	30	54	27	43	22
FAP_n4110_m1154467	4110	26	30	54	29	47	22
QAP_n144_m10672	144	202	223	2000*(1.1 -6)	233	2000*(2.5 -6)	1688
QAP_n196_m19619	196	207	220	674	244	2000*(3.3 -6)	2000*(2.3 -6)
QAP_n225_m25783	225	375	228	385	251	968	227
QAP_n256_m33302	256	236	253	382	259	2000*(1.7 -6)	2000*(1.2 -6)
QAP_n289_m42362	289	231	248	609	266	2000*(4.0 -6)	1357
QAP_n324_m53161	324	240	262	1001	275	2000*(1.9 -6)	1939
QAP_n400_m80828	400	263	258	1744	286	2000*(2.4 -6)	2000*(1.3 -6)
QAP_n441_m98152	441	254	269	1611	293	1455	2000*(1.8 -6)
QAP_n484_m118127	484	271	290	1763	299	1462	284
QAP_n625_m196598	625	274	304	1740	316	2000*(2.8 -6)	387
QAP_n676_m229877	676	283	297	885	323	2000*(2.7 -6)	342
QAP_n729_m267217	729	283	305	527	327	2000*(1.8 -6)	1646
QAP_n784_m308936	784	279	314	1626	333	2000*(2.0 -6)	1587
QAP_n900_m406843	900	302	323	1055	343	1393	330
QAP_n1225_m752813	1225	321	333	931	368	2000*(1.3 -6)	303
QAP_n1600_m1283258	1600	333	346	583	388	2000*(2.3 -6)	2000*(1.1 -6)
RAND_n300_m10K_p4	300	102	118	337	167	387	323
RAND_n300_m20K_p3	300	405	407	1411	690	2000*(1.1 -6)	2000
RAND_n300_m25K_p3	300	409	411	1579	1244	2000*(1.5 -6)	2000*(1.0 -6)
RAND_n400_m15K_p4	400	86	151	319	160	315	304
RAND_n400_m30K_p3	400	440	442	1176	939	1841	1866
RAND_n400_m40K_p3	400	405	407	1515	1090	2000*(1.5 -6)	2000*(1.3 -6)
RAND_n500_m20K_p4	500	109	149	372	153	485	459
RAND_n500_m30K_p3	500	313	262	944	449	1379	1104
RAND_n500_m40K_p3	500	353	355	1184	810	1747	1618
RAND_n500_m50K_p3	500	442	423	1297	1179	2000*(1.1 -6)	2000*(1.3 -6)
RAND_n600_m20K_p4	600	65	74	243	150	274	187
RAND_n600_m40K_p3	600	366	368	1011	522	1508	1542
RAND_n600_m50K_p3	600	323	325	1150	533	1755	1476
RAND_n600_m60K_p3	600	401	403	1327	851	2000*(1.6 -6)	2000*(1.2 -6)
RAND_n700_m50K_p3	700	327	330	989	558	1830	1570
RAND_n700_m70K_p3	700	356	358	1187	788	1843	1521
RAND_n700_m90K_p3	700	392	394	1420	1420	2000*(1.3 -6)	2000*(1.1 -6)
RAND_n800_m100K_p3	800	383	385	1285	971	1995	1688
RAND_n800_m110K_p3	800	378	380	1378	1182	2000*(1.2 -6)	2000*(1.1 -6)
RAND_n800_m70K_p3	800	274	276	1073	739	1911	1453
RAND_n900_m100K_p3	900	381	383	1190	849	1885	1831
RAND_n900_m140K_p3	900	368	379	1429	893	2000*(1.3 -6)	1712
RAND_n1K_m100K_p3	1000	280	282	1070	840	1657	1572
RAND_n1K_m150K_p3	1000	368	370	1351	895	2000*(1.2 -6)	1972

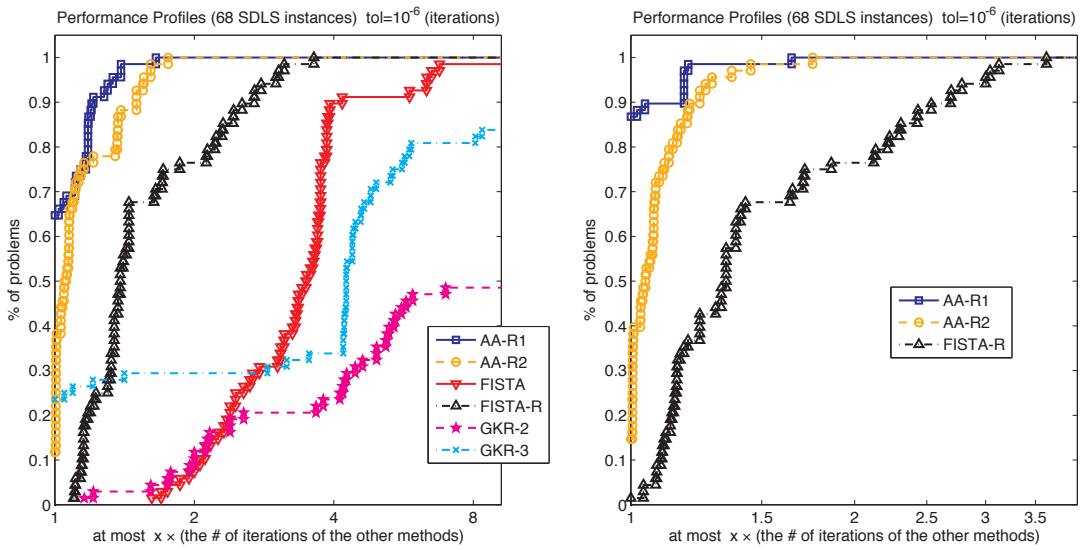

Figure 4.4.4: Number of iterations performance profiles for solving SDLS instances with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

instances randomly generated as in [32].

Given a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$, the NNLS problem is defined as

$$\min_{x\in\mathbb{R}^n} \left\{ \frac{1}{2} \|Ax - b\|^2 \ : \ x \geq 0 \right\}, \tag{4.4.4}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Letting f and h be defined as

$$f(x) = \frac{1}{2}\|Ax - b\|^2, \qquad h(x) = \delta_{\mathbb{R}^n_+}(x), \qquad \forall x \in \mathbb{R}^n,$$

where $\mathbb{R}^n_+$ is the cone of nonnegative vectors in $\mathbb{R}^n$, we can easily see that (4.4.4) is a special case of (1.2.1) with $\mathcal{X} = \mathbb{R}^n$ and ϕ given by (4.3.2).

Figures 4.4.5 and 4.4.6 plot the time and iteration performance profiles of all variants of Nesterov's method for solving this collection of random NNLS instances, respectively. Tables 4.4.5 and 4.4.6 report the time and number of iterations taken by each method, respectively.

4.5 Concluding remarks

We have observed in our computational experiments that AA-R2 quickly stops performing restarts, while AA-R1 periodically continues to perform restarts.

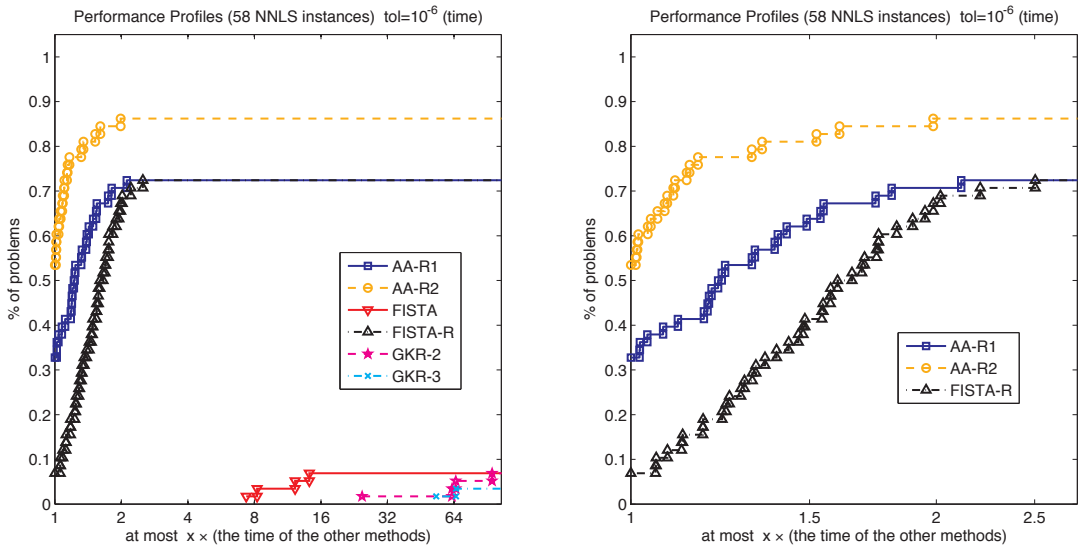

Figure 4.4.5: Time performance profiles for solving NNLS instances with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

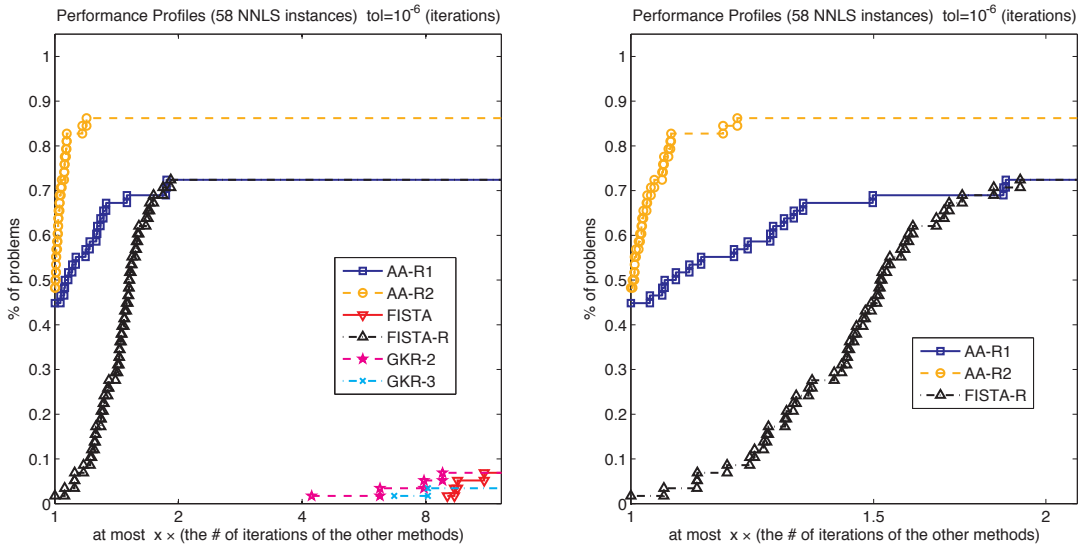

Figure 4.4.6: Number of iterations performance profiles for solving NNLS instances with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

Table 4.4.5: Time comparison of the methods on NNLS instances.

Problem		Time in seconds (accuracy less that 10^{-6})					
Instance	n	**AA-R1**	**AA-R2**	**FISTA**	**FISTA-R**	**GKR-2**	**GKR-3**
NNLS_n200_m10K	200	0.8	0.9	7.2*(2.3 -5)	1.4	25.7*(3.8 -5)	21.9*(3.3 -5)
NNLS_n200_m1K	200	1	0.6	2.2*(1.2 -5)	0.7	23.3*(1.9 -6)	16.8*(1.7 -5)
NNLS_n200_m20K	200	14.7*(2.2 -6)	0.8	6.2	1.1	46.4*(3.2 -6)	59.0*(4.8 -6)
NNLS_n200_m2K	200	0.2	0.2	1.9	0.3	14.7	14.7
NNLS_n200_m400	200	0.2	0.2	2.8	0.3	31.1*(6.7 -6)	12
NNLS_n200_m40K	200	12.8	10.7	52.3*(5.1 -6)	11.3	100.9*(1.3 -5)	87.5*(3.2 -4)
NNLS_n200_m4K	200	1.7	1.4	3.0*(1.4 -4)	1.8	33.0*(1.1 -5)	40.5*(3.3 -4)
NNLS_n200_m600	200	0.9	0.8	1.3*(1.7 -5)	0.8	17.2*(6.2 -6)	21.1*(5.6 -5)
NNLS_n200_m6K	200	0.5	0.5	3.3*(1.7 -5)	0.6	19.6*(1.3 -5)	20.8*(5.0 -6)
NNLS_n200_m800	200	0.8	0.8	2.4*(7.7 -5)	1	20.1*(8.0 -6)	23.7*(5.4 -5)
NNLS_n200_m8K	200	1.8	5.4*(1.1 -6)	5.0*(3.1 -4)	3.2	36.5*(1.6 -5)	31.7*(4.1 -4)
NNLS_n400_m10K	400	3.9	2.6	13.7*(3.0 -4)	4.8	35.8*(1.1 -4)	59.7*(1.9 -4)
NNLS_n400_m1K	400	0.4	0.4	2.7*(2.1 -6)	0.5	26.4	23.0*(7.0 -6)
NNLS_n400_m20K	400	8.5	7.7	29.3*(3.7 -4)	7.6	118.3*(1.5 -5)	105.6*(3.0 -4)
NNLS_n400_m2K	400	1.4	1.5	4.0*(1.4 -4)	1.5	41.2*(6.5 -6)	41.0*(1.1 -4)
NNLS_n400_m40K	400	61.7*(1.1 -6)	11.9	55.7*(7.9 -5)	23.2	284.7*(2.0 -5)	101.7*(1.9 -4)
NNLS_n400_m4K	400	0.9	0.8	7.1*(7.9 -6)	0.8	44.2*(1.9 -6)	44.3*(5.5 -6)
NNLS_n400_m6K	400	8.7*(1.0 -6)	1.2	6.6*(4.5 -5)	1.9	24.7*(2.0 -6)	35.2*(4.1 -5)
NNLS_n400_m800	400	0.3	0.3	4.1*(1.2 -6)	0.5	30.8	31.4*(4.6 -6)
NNLS_n400_m8K	400	2.8	4.6	12.3*(2.6 -4)	8.4*(1.1 -6)	43.2*(1.2 -4)	60.1*(2.9 -4)
NNLS_n600_m10K	600	3.1	3.6	17.6*(8.4 -5)	17.2*(1.9 -6)	47.9*(6.8 -6)	45.5*(5.6 -5)
NNLS_n600_m20K	600	28.6*(2.2 -6)	3.2	45.6	40.8*(1.5 -6)	122.8*(1.8 -5)	93.0*(4.9 -6)
NNLS_n600_m2K	600	1.1	1.1	3.3*(6.1 -6)	1.8	32.2*(2.6 -6)	28.7*(5.0 -5)
NNLS_n600_m40K	600	79.5*(4.6 -6)	65.1*(2.6 -6)	88.2*(5.8 -5)	10.5	207.4*(4.5 -5)	263.6*(3.3 -5)
NNLS_n600_m4K	600	1.8	1.2	5.4*(6.6 -6)	1.3	41.0*(2.0 -6)	31.0*(3.5 -5)
NNLS_n600_m6K	600	1.2	1.1	13.5*(7.5 -5)	1.7	39.5*(4.6 -5)	55.5*(6.0 -5)
NNLS_n600_m8K	600	1.6	1.8	18.0*(3.0 -5)	2.5	44.8*(2.8 -5)	51.2*(1.8 -5)
NNLS_n800_m10K	800	2.4	3.2	33.9*(2.2 -5)	3.5	105.8*(2.4 -6)	101.1*(1.3 -5)
NNLS_n800_m20K	800	9.5	6.8	38.8*(6.6 -5)	12.9	104.8*(1.8 -5)	91.9*(2.1 -4)
NNLS_n800_m2K	800	1.2	1.2	4.3*(1.2 -4)	1.7	29.1*(4.2 -5)	28.5*(7.8 -5)
NNLS_n800_m40K	800	70	33.1	93.7*(1.4 -4)	72.0*(5.6 -6)	360.0*(3.4 -5)	222.6*(3.9 -4)
NNLS_n800_m4K	800	3.1	2.3	12.2*(8.3 -5)	5.9	29.0*(5.3 -6)	63.1*(7.1 -5)
NNLS_n800_m6K	800	1.5	1.7	11.2*(1.2 -5)	2.5	49.1*(4.0 -6)	44.3*(1.4 -5)
NNLS_n800_m8K	800	2.7	5.4	29.7*(3.1 -6)	22.4*(1.1 -6)	67.1	91.9*(1.3 -5)
NNLS_n1K_m10K	1000	23.3*(1.8 -6)	3.5	31.2*(4.7 -5)	5.1	132.0*(4.0 -5)	97.1*(3.9 -5)
NNLS_n1K_m20K	1000	11.4	11.2	59.7*(7.5 -5)	50.0*(1.1 -6)	244.5*(6.3 -5)	166.8*(7.1 -5)
NNLS_n1K_m2K	1000	1.7	1.2	4.3*(2.0 -4)	2.1	45.1*(1.8 -5)	26.2*(1.3 -4)
NNLS_n1K_m40K	1000	103.6*(3.5 -6)	95.3*(1.1 -6)	143.5*(3.1 -4)	137.5*(1.8 -6)	331.9*(3.1 -5)	276.9*(4.0 -4)
NNLS_n1K_m4K	1000	1.9	1.6	7.6*(2.5 -5)	2.2	59.4*(4.7 -6)	35.5*(3.5 -5)
NNLS_n1K_m6K	1000	1.9	2.8	17.5*(1.1 -5)	2.1	56.6*(4.6 -6)	50.9*(5.3 -6)
NNLS_n1K_m8K	1000	7.2	5.2	18.7*(5.2 -5)	6.5	82.3*(1.3 -5)	74.2*(1.4 -4)
NNLS_n2K_m10K	2000	14.2	11.5	71.8*(5.1 -5)	18.1	139.7*(7.8 -6)	133.3*(1.0 -4)
NNLS_n2K_m20K	2000	89.2*(1.2 -6)	24.7	108.4*(2.3 -5)	54.7	371.2*(2.3 -5)	236.4*(2.4 -5)
NNLS_n2K_m40K	2000	209.7*(5.5 -6)	29.7	221.7*(2.2 -5)	60	573.2*(1.2 -5)	800.8*(1.2 -5)
NNLS_n2K_m4K	2000	7.5	6.9	23.9*(3.1 -5)	6.4	79.2*(2.4 -5)	91.5*(2.6 -5)
NNLS_n2K_m6K	2000	19.7	12.8	27.2*(8.4 -5)	19.8	124.6*(3.7 -5)	131.4*(6.2 -5)
NNLS_n2K_m8K	2000	3.6	4.9	31.7*(1.3 -5)	37.3*(1.6 -6)	127.8*(8.2 -6)	82.6*(1.1 -5)
NNLS_n4K_m10K	4000	94.8*(1.6 -6)	23.7	88.1*(4.9 -5)	31.5	468.4*(4.1 -5)	351.1*(3.6 -5)
NNLS_n4K_m20K	4000	53.6	157.3*(1.2 -6)	180.8*(8.1 -5)	185.9*(1.3 -6)	561.3*(4.7 -5)	381.3*(5.8 -5)
NNLS_n4K_m40K	4000	472.0*(1.6 -6)	452.2*(1.6 -6)	435.9*(8.3 -5)	533.3*(2.9 -6)	1558.5*(1.3 -5)	1221.4*(1.4 -4)
NNLS_n4K_m8K	4000	21.9	16.5	67.3*(8.8 -5)	28.9	209.8*(2.1 -5)	145.3*(3.8 -5)
NNLS_n6K_m20K	6000	311.9*(2.5 -6)	79.7	370.5*(7.6 -5)	267.9*(1.2 -6)	1095.7*(9.9 -6)	1064.1*(6.5 -5)
NNLS_n6K_m40K	6000	643.4*(3.0 -6)	222	603.6*(1.2 -4)	620.1*(1.2 -6)	2254.7*(5.5 -5)	2521.3*(9.2 -5)
NNLS_n8K_m20K	8000	174.5	100.1	356.7*(6.6 -5)	198.9	1477.6*(3.7 -5)	1106.5*(7.3 -5)
NNLS_n8K_m40K	8000	109.6	592.2*(2.3 -6)	1068.5*(1.0 -5)	637.7*(1.4 -6)	3209.2*(6.2 -6)	2556.4*(2.0 -5)
NNLS_n10K_m20K	10000	469.2*(2.1 -6)	162.2	530.7*(1.1 -4)	478.3*(3.0 -6)	1623.3*(5.9 -5)	1410.5*(8.2 -5)
NNLS_n10K_m40K	10000	708.7*(2.6 -6)	712.7*(2.4 -6)	1039.4*(7.0 -5)	785.6*(2.8 -6)	3231.6*(2.6 -5)	2604.1*(1.3 -4)
NNLS_n20K_m40K	20000	1952.1*(1.2 -6)	2194.0*(1.8 -6)	1792.2*(7.5 -5)	2044.2*(3.4 -6)	6670.7*(3.9 -5)	4648.8*(6.9 -5)

Table 4.4.6: Number of iterations comparison of the methods on NNLS instances.

Problem		# of iterations (accuracy less that 10^{-6})					
Instance	n	**AA-R1**	**AA-R2**	**FISTA**	**FISTA-R**	**GKR-2**	**GKR-3**
NNLS_n200_m10K	200	325	330	2000*(2.3 -5)	553	2000*(3.8 -5)	2000*(3.3 -5)
NNLS_n200_m1K	200	836	447	2000*(1.2 -5)	689	2000*(1.9 -6)	2000*(1.7 -5)
NNLS_n200_m20K	200	2000*(2.2 -6)	138	1246	181	2000*(3.2 -6)	2000*(4.8 -6)
NNLS_n200_m2K	200	132	141	1240	199	1046	885
NNLS_n200_m400	200	168	171	1866	251	2000*(6.7 -6)	1360
NNLS_n200_m40K	200	936	738	2000*(5.1 -6)	824	2000*(1.3 -5)	2000*(3.2 -4)
NNLS_n200_m4K	200	979	758	2000*(1.4 -4)	1147	2000*(1.1 -5)	2000*(3.3 -4)
NNLS_n200_m600	200	809	641	2000*(1.7 -5)	800	2000*(6.2 -6)	2000*(5.6 -5)
NNLS_n200_m6K	200	230	246	2000*(1.7 -5)	323	2000*(1.3 -5)	2000*(5.0 -6)
NNLS_n200_m800	200	623	627	2000*(7.7 -5)	808	2000*(8.0 -6)	2000*(5.4 -5)
NNLS_n200_m8K	200	720	2000*(1.1 -6)	2000*(3.1 -4)	1320	2000*(1.6 -5)	2000*(4.1 -4)
NNLS_n400_m10K	400	647	669	2000*(3.0 -4)	943	2000*(1.1 -4)	2000*(1.9 -4)
NNLS_n400_m1K	400	291	275	2000*(2.1 -6)	370	1702	2000*(7.0 -6)
NNLS_n400_m20K	400	775	775	2000*(3.7 -4)	866	2000*(1.5 -5)	2000*(3.0 -4)
NNLS_n400_m2K	400	860	865	2000*(1.4 -4)	1049	2000*(6.5 -6)	2000*(1.1 -4)
NNLS_n400_m40K	400	2000*(1.1 -6)	391	2000*(7.9 -5)	556	2000*(2.0 -5)	2000*(1.9 -4)
NNLS_n400_m4K	400	344	363	2000*(7.9 -6)	404	2000*(1.9 -6)	2000*(5.5 -6)
NNLS_n400_m6K	400	2000*(1.0 -6)	387	2000*(4.5 -5)	673	2000*(2.0 -6)	2000*(4.1 -5)
NNLS_n400_m800	400	224	217	2000*(1.2 -6)	321	1908	2000*(4.6 -6)
NNLS_n400_m8K	400	744	706	2000*(2.6 -4)	2000*(1.1 -6)	2000*(1.2 -4)	2000*(2.9 -4)
NNLS_n600_m10K	600	432	444	2000*(8.4 -5)	2000*(1.9 -6)	2000*(6.8 -6)	2000*(5.6 -5)
NNLS_n600_m20K	600	2000*(2.2 -6)	194	1855	2000*(1.5 -6)	2000*(1.8 -5)	2000*(4.9 -6)
NNLS_n600_m2K	600	612	555	2000*(6.1 -6)	718	2000*(2.6 -6)	2000*(5.0 -5)
NNLS_n600_m40K	600	2000*(4.6 -6)	2000*(2.6 -6)	2000*(5.8 -5)	392	2000*(4.5 -5)	2000*(3.3 -5)
NNLS_n600_m4K	600	379	387	2000*(6.6 -6)	466	2000*(2.0 -6)	2000*(3.5 -5)
NNLS_n600_m6K	600	304	308	2000*(7.5 -5)	461	2000*(4.6 -5)	2000*(6.0 -5)
NNLS_n600_m8K	600	288	336	2000*(3.0 -5)	486	2000*(2.8 -5)	2000*(1.8 -5)
NNLS_n800_m10K	800	250	260	2000*(2.2 -5)	395	2000*(2.4 -6)	2000*(1.3 -5)
NNLS_n800_m20K	800	518	436	2000*(6.6 -5)	626	2000*(1.8 -5)	2000*(2.1 -4)
NNLS_n800_m2K	800	572	577	2000*(1.2 -4)	830	2000*(4.2 -5)	2000*(7.8 -5)
NNLS_n800_m40K	800	1431	768	2000*(1.4 -4)	2000*(5.6 -6)	2000*(3.4 -5)	2000*(3.9 -4)
NNLS_n800_m4K	800	769	774	2000*(8.3 -5)	1133	2000*(5.3 -6)	2000*(7.1 -5)
NNLS_n800_m6K	800	348	349	2000*(1.2 -5)	499	2000*(4.0 -6)	2000*(1.4 -5)
NNLS_n800_m8K	800	417	444	2000*(3.1 -6)	2000*(1.1 -6)	1760	2000*(1.3 -5)
NNLS_n1K_m10K	1000	2000*(1.8 -6)	352	2000*(4.7 -5)	372	2000*(4.0 -5)	2000*(3.9 -5)
NNLS_n1K_m20K	1000	411	433	2000*(7.5 -5)	2000*(1.1 -6)	2000*(6.3 -5)	2000*(7.1 -5)
NNLS_n1K_m2K	1000	749	500	2000*(2.0 -4)	958	2000*(1.8 -5)	2000*(1.3 -4)
NNLS_n1K_m40K	1000	2000*(3.5 -6)	2000*(1.1 -6)	2000*(3.1 -4)	2000*(1.8 -6)	2000*(3.1 -5)	2000*(4.0 -4)
NNLS_n1K_m4K	1000	345	320	2000*(2.5 -5)	533	2000*(4.7 -6)	2000*(3.5 -5)
NNLS_n1K_m6K	1000	263	278	2000*(1.1 -5)	356	2000*(4.6 -6)	2000*(5.3 -6)
NNLS_n1K_m8K	1000	718	547	2000*(5.2 -5)	870	2000*(1.3 -5)	2000*(1.4 -4)
NNLS_n2K_m10K	2000	705	580	2000*(5.1 -5)	889	2000*(7.8 -6)	2000*(1.0 -4)
NNLS_n2K_m20K	2000	2000*(1.2 -6)	414	2000*(2.3 -5)	521	2000*(2.3 -5)	2000*(2.4 -5)
NNLS_n2K_m40K	2000	2000*(5.5 -6)	286	2000*(2.2 -5)	449	2000*(1.2 -5)	2000*(1.2 -5)
NNLS_n2K_m4K	2000	804	715	2000*(3.1 -5)	897	2000*(2.4 -5)	2000*(2.6 -5)
NNLS_n2K_m6K	2000	828	850	2000*(8.4 -5)	1193	2000*(3.7 -5)	2000*(6.2 -5)
NNLS_n2K_m8K	2000	262	313	2000*(1.3 -5)	2000*(1.6 -6)	2000*(8.2 -6)	2000*(1.1 -5)
NNLS_n4K_m10K	4000	2000*(1.6 -6)	543	2000*(4.9 -5)	826	2000*(4.1 -5)	2000*(3.6 -5)
NNLS_n4K_m20K	4000	668	2000*(1.2 -6)	2000*(8.1 -5)	2000*(1.3 -6)	2000*(4.7 -5)	2000*(5.8 -5)
NNLS_n4K_m40K	4000	2000*(1.6 -6)	2000*(1.6 -6)	2000*(8.3 -5)	2000*(2.9 -6)	2000*(1.3 -5)	2000*(1.4 -4)
NNLS_n4K_m8K	4000	580	591	2000*(8.8 -5)	929	2000*(2.1 -5)	2000*(3.8 -5)
NNLS_n6K_m20K	6000	2000*(2.5 -6)	490	2000*(7.6 -5)	2000*(1.2 -6)	2000*(9.9 -6)	2000*(6.5 -5)
NNLS_n6K_m40K	6000	2000*(3.0 -6)	695	2000*(1.2 -4)	2000*(1.2 -6)	2000*(5.5 -5)	2000*(9.2 -5)
NNLS_n8K_m20K	8000	964	723	2000*(6.6 -5)	953	2000*(3.7 -5)	2000*(7.3 -5)
NNLS_n8K_m40K	8000	334	2000*(2.3 -6)	2000*(1.0 -5)	2000*(1.4 -6)	2000*(6.2 -6)	2000*(2.0 -5)
NNLS_n10K_m20K	10000	2000*(2.1 -6)	744	2000*(1.1 -4)	2000*(3.0 -6)	2000*(5.9 -5)	2000*(8.2 -5)
NNLS_n10K_m40K	10000	2000*(2.6 -6)	2000*(2.4 -6)	2000*(7.0 -5)	2000*(2.8 -6)	2000*(2.6 -5)	2000*(1.3 -4)
NNLS_n20K_m40K	20000	2000*(1.2 -6)	2000*(1.8 -6)	2000*(7.5 -5)	2000*(3.4 -6)	2000*(3.9 -5)	2000*(6.9 -5)

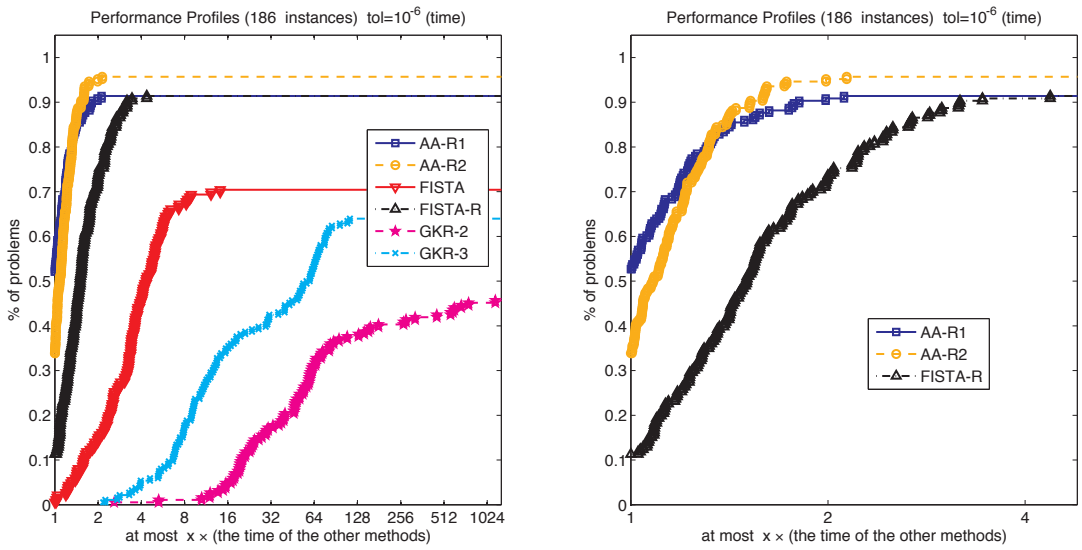

Figure 4.5.1: Time performance profiles for solving all the three problem classes with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

Figures 4.5.1 and 4.5.2 plot the time and iteration performance profiles of all variants of Nesterov's method on the collection of instances obtained by combining all the three instance classes described in the previous subsections. From these plots we can see that the overall performances of AA-R1 and AA-R2 are very close to one another, with AA-R2 turning out to be more robust, as it solves more problems to the specified accuracy 10^{-6} than any other of the variants tested. Finally, we can see that AA-R1 and AA-R2 are the two fastest variants among the six ones used in this benchmark in terms of both time and number of iterations.

Acknowledgements. We want to thank Elizabeth W. Karas for generously providing us with the code of the algorithms in [20].

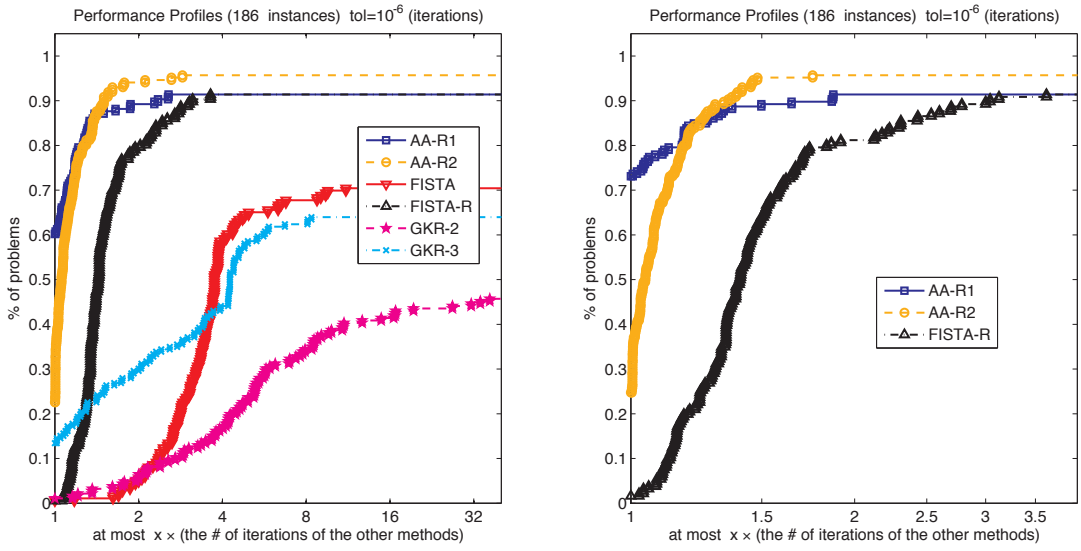

Figure 4.5.2: Iterations performance profiles for solving all the three problem classes with accuracy $\bar{\epsilon} = 10^{-6}$. On the left we include all the methods and on the right we include the three fastest variants, namely: AA-R1, AA-R2 and FISTA-R.

Chapter V

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis we have investigated new block-decomposition and accelerated gradient methods for large-scale optimization. In terms of developing BD methods for large-scale conic programming, we have:

- presented an adaptive BD framework, and corresponding iteration-complexity results, in the context of a block-structured monotone inclusion problem which provides a blueprint to derive all the BD methods proposed in this thesis;
- introduced a speed-up refinement on the BD framework that, based on the HPE condition, adaptively (aggressively) chooses the size of extragradient step;
- introduced important speed-up refinements on the implementation of the BD methods that uses scaled inner products to balance the blocks by properly initializing and/or dynamically updating the scaling parameters;
- developed and implemented an exact BD method for standard form conic programming problems as in (1.1.1), and presented numerical results showing that it generally outperforms the methods of [30] and [68];
- developed and implemented an exact BD method for composite convex optimization, and presented numerical results showing that it generally outperforms the variant of the BP method in [67], as well as the methods of [30] and [68], in a benchmark that included the same classes of large-scale graph-related SDP problems tested in [67];
- developed and implemented an inexact BD method for solving standard form conic programming problems which avoids computations of exact projections onto the manifold defined by the affine constraints and, as a result, is able to handle extra large-scale SDP instances, several of which cannot be solved by other existing methods.

On the other hand, in terms of developing accelerated gradient methods for large-scale convex optimization, we have:

- presented a generic accelerated framework for convex optimization which outlines sufficient conditions at each iteration to guarantee $\mathcal{O}(1/k^2)$ convergence on the functional gap for constant stepsizes;
- introduced a procedure that adaptively and aggressively chooses certain acceleration parameters in order to substantially improve the practical performance of accelerated methods without compromising their theoretical iteration-complexity;
- introduced restart schemes on the acceleration parameters that adaptively force descent iterations as the method progresses which led to significant performance improvements in our numerical experiments;
- developed and implemented adaptive accelerated gradient methods for convex optimization problems and presented numerical results demonstrating that they perform quite well compared to other variants of Nesterov's method proposed earlier in the literature.

It should be mentioned that all the research and results of this thesis involved the implementation of (open-source) optimization algorithms which can be found at `http://www.isye.gatech.edu/~cod3/COrtiz/software/`.

The following topics are worth mentioning for future studies:

- recognize relevant applications that explicitly benefits from a BD method designed for more than two blocks since, to the best of our knowledge, a two-block design has been sufficient for most applications in convex optimization;
- develop efficient variants of the proposed BD methods that incorporate the use of low-rank representations of the matrix variables to satisfy the increasing demand for handling larger instances;
- develop specialized implementations of efficient variants of the proposed BD methods that directly handle convex optimization problems as in (2.3.1) where $h_i \neq \delta_C$, in

particular, the cases that include the minimization of ℓ_1, ℓ_2, and/or nuclear norms in view of their multiple applications in imaging, statistical machine learning and compressed sensing;

- study the performance of the proposed adaptive accelerated methods with different restart schemes (e.g., one that adaptively forces a decrease on the projected gradient as the method progresses), in view that these schemes played a key role in the practical performance of our methods.

Appendix A

TECHNICAL RESULTS

In this appendix we present and review some technical results.

A.1 Expectation of the norm of a linear mapping

Lemma A.1 (Theorem 2.2 in [3]). *Given a self adjoint positive definite linear mapping $\mathcal{U} : \mathcal{Y} \to \mathcal{Y}$ and a random vector $y \in \mathcal{Y}$ uniformly distributed on a ball, we have that*

$$\mathbb{E}\left(\frac{\left\|\mathcal{U}^{1/2}y\right\|^2}{\|\mathcal{U}\|\,\|y\|^2}\right) = \frac{1}{\sigma_m}\frac{\sum\sigma_i}{m} \leq 1$$

where $\sigma_1 \leq \cdots \leq \sigma_m$ are the eigenvalues values of $\mathcal{U}$.

A.2 Ergodic convergence results of Algorithm 2.2

This section derives an ergodic iteration-complexity bound for Algorithm 2.2.

The weak transportation formula for the ε-subdifferential is stated in the following result.

Proposition A.2 (Proposition 1.2.10 in [26]). *Suppose that $f : \mathcal{Z} \rightrightarrows \bar{\mathbb{R}}$ is a closed proper convex function. Let $z^i, v^i \in \mathcal{Z}$ and $\varepsilon_i, \alpha_i \in \mathbb{R}_+$, for $i = 1, \ldots, k$, be such that*

$$v^i \in \partial_{\varepsilon_i} f(z^i), \quad i = 1, \ldots, k, \qquad \sum_{i=1}^k \alpha_i = 1,$$

and define

$$z_a := \sum_{i=1}^k \alpha_i z^i, \qquad v_a := \sum_{i=1}^k \alpha_i v^i,$$

$$\varepsilon_a := \sum_{i=1}^k \alpha_i \left[\varepsilon_i + \left\langle z^i - z_a, v^i - v_a\right\rangle_{\mathcal{Z}}\right] = \sum_{i=1}^k \alpha_i \left[\varepsilon_i + \left\langle z^i - z_a, v^i\right\rangle_{\mathcal{Z}}\right].$$

Then, $\varepsilon_a \geq 0$ and $v_a \in \partial_{\varepsilon_a} f(z_a)$.

The following result derives an ergodic iteration-complexity bound for Algorithm 2.2.

Theorem A.3. *Consider the sequences* $\{(x^k, y^k)\}$, $\{(\tilde{x}^k, \tilde{y}^k)\}$, $\{(v_1^k, v_2^k)\}$ *and* $\{\varepsilon_k\}$ *generated by Algorithm 2.2, and the sequences* $\{\tilde{c}^k\}$ *and* $\{\tilde{d}^k\}$ *defined in* (2.3.18). *For every* $k \in \mathbb{N}$, *define*

$$\Lambda_k := \sum_{i=1}^{k} \lambda_i, \qquad (\tilde{x}^{a,k}, \tilde{y}^{a,k}) := \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i (\tilde{x}^i, \tilde{y}^i),$$

$$(v_1^{a,k}, v_2^{a,k}) := \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i (v_1^k, v_2^k), \qquad (\tilde{c}^{a,k}, \tilde{d}^{a,k}) := \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i (\tilde{c}^i, \tilde{d}^i)$$

and

$$\varepsilon_k^{1,a} := \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i [\varepsilon_k + \langle \theta^{-1} \tilde{c}^i, \tilde{x}^i - \tilde{x}^{a,k} \rangle], \tag{A.2.1a}$$

$$\varepsilon_k^{2,a} := \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i \langle \tilde{d}^i, \tilde{y}^i - \tilde{y}^{a,k} \rangle, \tag{A.2.1b}$$

$$\varepsilon_k^{a} := \varepsilon_k^{1,a} + \varepsilon_k^{2,a}. \tag{A.2.1c}$$

Then, for every $k \in \mathbb{N}$,

$$\begin{aligned} (\theta^{-1} v_1^{a,k}, v_2^{a,k}) &\in \left[\partial_{\varepsilon_k^{1,a}} \left(f + h_1 + \langle \tilde{y}^{a,k}, \cdot \rangle\right)(\tilde{x}^{a,k})\right] \times \left[\partial_{\varepsilon_k^{2,a}} \left(h_2^* - \langle \tilde{y}^{a,k}, \cdot \rangle\right)(\tilde{y}^{a,k})\right] \\ &\subseteq \partial_{\varepsilon_k^a} [\mathcal{L}(\cdot, \tilde{y}^{a,k}) - \mathcal{L}(\tilde{x}^{a,k}, \cdot)](\tilde{x}^{a,k}, \tilde{y}^{a,k}) \end{aligned} \tag{A.2.2}$$

and

$$\sqrt{\theta^{-1} \|v_1^{a,k}\|^2 + \|v_2^{a,k}\|^2} \leq \max\left\{\frac{1}{\sigma}, \frac{\sqrt{\theta} L}{\sigma_1^2}\right\} \left(\frac{2\sqrt{\theta}}{k}\right) \sqrt{\theta^{-1} d_{x,0}^2 + d_{y,0}^2}, \tag{A.2.3}$$

$$\varepsilon_k^a \leq \max\left\{1, \frac{\sqrt{\theta} L \sigma}{\sigma_1^2}\right\} \left[\frac{8\sqrt{\theta}}{(1-\sigma_1)k}\right] \left(\theta^{-1} d_{x,0}^2 + d_{y,0}^2\right), \tag{A.2.4}$$

where $d_{x,0}$ *and* $d_{y,0}$ *are defined in* (2.3.23).

Proof. Let $k \in \mathbb{N}$ be given. Note that by (2.3.27) and the definition of $\langle \cdot, \cdot \rangle_\theta$, we have

$$\varepsilon_k^{1,a} = \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i [\varepsilon_k + \langle \tilde{c}^i, \tilde{x}^i - \tilde{x}^{a,k} \rangle_\theta], \quad \varepsilon_k^{2,a} = \Lambda_k^{-1} \sum_{i=1}^{k} \lambda_i \langle \tilde{d}^i, \tilde{y}^i - \tilde{y}^{a,k} \rangle.$$

Then, in view of Lemma 2.13 and Theorem 2.4 in [35], we have

$$\|F(\tilde{x}^{a,k}, \tilde{y}^{a,k}) + (\tilde{c}^{a,k}, \tilde{d}^{a,k})\|_{\theta,1} \leq 2\frac{d_0^\theta}{\Lambda_k}, \qquad \varepsilon_k^a = \varepsilon_k^{1,a} + \varepsilon_k^{2,a} \leq \left(\frac{8\sigma}{1-\sigma_1}\right) \frac{(d_0^\theta)^2}{\Lambda_k}.$$

Hence, it follows from the above relations, Lemma 2.13(d) and the fact that $\lambda_k \geq \tilde{\lambda}$, that

$$\|(v_1^{a,k}, v_2^{a,k})\|_{\theta,1} = \|F(\tilde{x}^{a,k}, \tilde{y}^{a,k}) + (\tilde{c}^{a,k}, \tilde{d}^{a,k})\|_{\theta,1} \leq 2\frac{d_0^\theta}{\Lambda_k} \leq 2\frac{d_0^\theta}{k\tilde{\lambda}}, \ \varepsilon_k^a \leq \left(\frac{8\sigma}{1-\sigma_1}\right) \frac{(d_0^\theta)^2}{k\tilde{\lambda}}.$$

Using the definition of $\|(\cdot,\cdot)\|_{\theta,1}$, (2.3.22) and the definition of $\tilde{\lambda}$ in (2.3.13), we easily see that the above two inequalities imply (A.2.3) and (A.2.4). Now, (2.3.20), (2.3.21), (2.3.27), (A.2.1) and Proposition A.2 imply that

$$\theta^{-1}v_1^{a,k} \in \partial_{\varepsilon_k^{1,a}}(f+h_1)(\tilde{x}^{a,k}) + \tilde{y}^{a,k}, \qquad v_2^{a,k} \in \partial_{\varepsilon_k^{2,a}}(h_2^*)(\tilde{y}^{a,k}) - \tilde{x}^{a,k}.$$

and hence that

$$\theta^{-1}v_1^{a,k} \in (\partial_{x,\varepsilon_k^{1,a}}\mathcal{L})(\tilde{x}^{a,k},\tilde{y}^{a,k}), \qquad v_2^{a,k} \in (\partial_{y,\varepsilon_k^{2,a}}\mathcal{L})(\tilde{x}^{a,k},\tilde{y}^{a,k}).$$

The above four inclusions are easily seen to imply (A.2.2). □

A.3 Iteration-complexity of the CG method

Let a self adjoint positive definite linear operator $\tilde{\mathcal{Q}} : \mathcal{Y} \to \mathcal{Y}$ and $\tilde{q} \in \mathcal{Y}$ be given. Consider the problem of finding $\tilde{\Delta} \in \mathcal{Y}$ such that

$$\left\| \tilde{\mathcal{Q}}\tilde{\Delta} - \tilde{q} \right\| \leq \delta \|\tilde{\Delta}\|. \tag{A.3.1}$$

The following result estimates the number of iterations for the CG method to obtain a solution of (A.3.1). (See for example Algorithm 5.2 of [50] for a nice description of the CG method.)

Proposition A.4. *Let $\tilde{\Delta}^i$ be the ith iterate generated by the CG method applied to the linear system $\tilde{\mathcal{Q}}\tilde{\Delta} = \tilde{q}$ and with initial point $\tilde{\Delta}^0 = 0$. Then, $\tilde{\Delta}^i$ satisfies* (A.3.1) *for every*

$$i \geq \left\lceil \frac{1}{2}\sqrt{\kappa}\ln\left[2\sqrt{\kappa}\left(1+\frac{\|\tilde{\mathcal{Q}}\|}{\delta}\right)\right]\right\rceil, \tag{A.3.2}$$

where $\kappa = \kappa(\tilde{\mathcal{Q}})$.

Proof. Let $\tilde{\Delta}^*$ be the solution of the linear system $\tilde{\mathcal{Q}}\tilde{\Delta} = \tilde{q}$ and define $e^i := \tilde{\Delta}^i - \tilde{\Delta}^*$ for every $i \geq 0$. For a given $\epsilon > 0$, it is well-known (see for example (5.36) of [50]) that

$$\langle e^i, \tilde{\mathcal{Q}}e^i\rangle \leq 4\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2i}\langle e^0, \tilde{\mathcal{Q}}e^0\rangle \qquad \forall i \geq 0.$$

Using the latter inequality, it is easy to see that

$$\|e^i\| \leq \epsilon\|e^0\| \qquad \forall i \geq \left\lceil \tfrac{1}{2}\sqrt{\kappa}\ln\tfrac{2\sqrt{\kappa}}{\epsilon}\right\rceil.$$

The latter observation with

$$\epsilon = \frac{\delta}{\delta + \|\tilde{\mathcal{Q}}\|} \tag{A.3.3}$$

and the fact that $\epsilon\|\tilde{\mathcal{Q}}\| = \delta\,[1-\epsilon]$, then imply that, for every i satisfying (A.3.2), we have

$$\begin{aligned}\left\|\tilde{\mathcal{Q}}\tilde{\Delta}^i - \tilde{q}\right\| &= \left\|\tilde{\mathcal{Q}}\tilde{\Delta}^i - \tilde{\mathcal{Q}}\tilde{\Delta}^*\right\| \leq \left\|\tilde{\mathcal{Q}}\right\| \|e^i\| \leq \epsilon \left\|\tilde{\mathcal{Q}}\right\| \|e^0\| = \delta\left[\|e^0\| - \epsilon\|e^0\|\right] \\ &\leq \delta\left[\|e^0\| - \|e^i\|\right] \leq \delta\|e^0 - e^i\| = \delta\|\tilde{\Delta}^0 - \tilde{\Delta}^i\| = \delta\|\tilde{\Delta}^i\|,\end{aligned}$$

where the last equality is due to the assumption that $\tilde{\Delta}^0 = 0$. □

A.4 The Ramsey multiplicity problem

Given a graph G and $t \in \mathbb{N}$, let K_t denote the complete graph on t vertices, $k_t(G)$ denote the number of cliques of size t in G, and $\bar{G}$ denote the complement of G. Given $n \in \mathbb{N}$, define $k_t(n)$ as the minimum number of monochromatic copies of K_t in an improper 2-edge-coloring of K_n, i.e., $k_t(n) := \min\{k_t(G) + k_t(\bar{G}) : |G| = n\}$. If n is sufficiently large compared to t, it follows from Ramsey's theorem that $k_t(n) > 0$. Letting

$$c_t := \lim_{n\to\infty} \frac{k_t(n)}{\binom{n}{t}},$$

the problem of determining c_t is known as the Ramsey Multiplicity Problem (RMP). Moreover, a lower bound on c_t can be found by solving an SDP of the form

$$\begin{aligned}\min\ & -x \\ \text{s.t}\ & \mathcal{A}(X) + xe \leq b, \\ & X \in \mathcal{S}^n_+,\ x \in \mathbb{R},\end{aligned}$$

where $b \in \mathbb{R}^m_+$, $e = (1,\ldots,1)^T \in \mathbb{R}^m$ and $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$ is a linear operator (see [49] for details).

The SDP instances from this problem class used in Section 2.3.4 were made available to us by S. Nieß.

REFERENCES

[1] Beck, A. and Teboulle, M., "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, pp. 183–202, Mar. 2009.

[2] Benson, S. J., Ye, Y., and Zhang, X., "Solving large-scale sparse semidefinite programs for combinatorial optimization," *SIAM Journal on Optimization*, vol. 10, pp. 443–461, Jan. 2000.

[3] Bottcher, A. and Grudsky, S., "The norm of the product of a large matrix and a random vector.," *Electronic Journal of Probability [electronic only]*, vol. 8, pp. Paper No. 7, 29 p., electronic only–Paper No. 7, 29 p., electronic only, 2003.

[4] Burachik, R. S., Iusem, A. N., and Svaiter, B. F., "Enlargement of monotone operators with applications to variational inequalities," *Set-Valued Analysis*, vol. 5, pp. 159–180, 1997. 10.1023/A:1008615624787.

[5] Burachik, R. S. and Svaiter, B. F., "Maximal monotone operators, convex functions and a special family of enlargements," *Set-Valued Analysis*, vol. 10, pp. 297–316, 2002. 10.1023/A:1020639314056.

[6] Burer, S. and Choi, C., "Computational enhancements in low-rank semidefinite programming," *Optimization Methods and Software*, vol. 21, no. 3, pp. 493–512, 2006.

[7] Burer, S. and Monteiro, R. D. C., "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Mathematical Programming*, vol. 95, no. 2, pp. 329–357, 2003.

[8] Burer, S. and Monteiro, R. D. C., "Local minima and convergence in low-rank semidefinite programming," *Mathematical Programming*, vol. 103, no. 3, pp. 427–444, 2005.

[9] Burer, S., Monteiro, R. D. C., and Zhang, Y., "A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs," *Mathematical Programming*, vol. 95, pp. 359–379, 2003. 10.1007/s10107-002-0353-7.

[10] Burer, S. and Vandenbussche, D., "Solving lift-and-project relaxations of binary integer programs," *SIAM Journal on Optimization*, vol. 16, no. 3, pp. 726–750, 2006.

[11] Chambolle, A. and Pock, T., "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, pp. 120–145, 2011. 10.1007/s10851-010-0251-1.

[12] d'Aspremont, A., "Smooth optimization with approximate gradient," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1171–1183, 2008.

[13] Devolder, O., Glineur, F., and Nesterov, Y., "First-order methods of smooth convex optimization with inexact oracle," core discussion paper 2011/02, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, December 2010.

[14] Devolder, O., Glineur, F., and Nesterov, Y., "Double smoothing technique for large-scale linearly constrained convex optimization," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 702–727, 2012.

[15] Dolan, E. D. and Moré, J. J., "Benchmarking optimization software with performance profiles," Jan. 2002.

[16] Gabay, D. and Mercier, B., "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers Mathematics with Applications*, vol. 2, no. 1, pp. 17 – 40, 1976.

[17] Gillberg, J. and Hansson, A., "Polynomial complexity for a Nesterov-Todd potential-reduction method with inexact search directions : Examples related to the KYP lemma," Tech. Rep. 2511, Linköping University, The Institute of Technology, 2003.

[18] Glowinski, R. and Marrocco, A., "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par penalisation-dualité, d'une classe de problèmes de dirichlet non linéaires," *RAIRO Anal. Numér.*, vol. 2, pp. 41 – 76, 1975.

[19] Gonzaga, C. C. and Karas, E. W., "Fine tuning Nesterov's steepest descent algorithm for differentiable convex programming," *Mathematical Programming*, pp. 1–26, 2012.

[20] Gonzaga, C. C., Karas, E. W., and Rossetto, D. R., "An optimal algorithm for constrained differentiable convex optimization," *Optimization-online preprint 3053*, pp. 1–16, 2011.

[21] Helmberg, C. and Rendl, F., "A spectral bundle method for semidefinite programming," *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 673–696, 2000.

[22] Joshi, S. and Boyd, S., "An efficient method for large-scale gate sizing," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 55, pp. 2760–2773, Oct 2008.

[23] Kim, S.-J., Koh, K., Lustig, M., Boyd, S., and Gorinevsky, D., "An interior-point method for large-scale ℓ_1-regularized least squares," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, pp. 606–617, Dec 2007.

[24] Koh, K., Kim, S.-J., and Boyd, S., "An interior-point method for large-scale ℓ_1-regularized logistic regression.," *Journal of Machine learning research*, vol. 8, no. 7, 2007.

[25] Lan, G., Lu, Z., and Monteiro, R., "Primal-dual first-order methods with iteration-complexity for cone programming," *Mathematical Programming*, vol. 126, no. 1, pp. 1–29, 2011.

[26] Lemaréchal, C., "Extensions diverses des méthodes de gradient et applications," tech. rep., Thèse d'Etat, Université de Paris IX, 1980.

[27] Liu, Z. and Vandenberghe, L., "Interior-point method for nuclear norm approximation with application to system identification," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1235–1256, 2010.

[28] Lu, Z., Nemirovski, A., and Monteiro, R. D. C., "Large-scale semidefinite programming via a saddle point mirror-prox algorithm," *Mathematical programming*, vol. 109, no. 2-3, pp. 211–237, 2007.

[29] Ma, S., Yin, W., Zhang, Y., and Chakraborty, A., "An efficient algorithm for compressed mr imaging using total variation and wavelets," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1 –8, june 2008.

[30] Malick, J., Povh, J., Rendl, F., and Wiegele, A., "Regularization methods for semidefinite programming," *SIAM J. on Optimization*, vol. 20, pp. 336–356, Apr. 2009.

[31] Martinet, B., "Brève communication. régularisation d'inéquations variationnelles par approximations successives," *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, vol. 4, no. R3, pp. 154–158, 1970.

[32] Merritt, M. and Zhang, Y., "Interior-point gradient method for large-scale totally nonnegative least squares problems," *Journal of Optimization Theory and Applications*, vol. 126, pp. 191–202, 2005. 10.1007/s10957-005-2668-z.

[33] Monteiro, R. D. C., Ortiz, C., and Svaiter, B. F., "A first-order block-decomposition method for solving two-easy-block structured semidefinite programs," *Mathematical Programming Computation*, pp. 1–48, 2013.

[34] Monteiro, R. D. C., Ortiz, C., and Svaiter, B. F., "An inexact block-decomposition method for extra large-scale conic semidefinite programming," *Optimization-online preprint 4158*, pp. 1–21, 2013.

[35] Monteiro, R. D. C., Ortiz, C., and Svaiter, B. F., "Implementation of a block-decomposition algorithm for solving large-scale conic semidefinite programming problems," *Computational Optimization and Applications*, vol. 57, no. 1, pp. 45–69, 2014.

[36] Monteiro, R. D. C. and Svaiter, B. F., "On the complexity of the hybrid proximal extragradient method for the iterates and the ergodic mean," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 2755–2787, 2010.

[37] Monteiro, R. D. C. and Svaiter, B. F., "Complexity of variants of Tseng's modified F-B splitting and korpelevich's methods for hemivariational inequalities with applications to saddle-point and convex optimization problems," *SIAM Journal on Optimization*, vol. 21, no. 4, pp. 1688–1720, 2011.

[38] Monteiro, R. D. C. and Svaiter, B. F., "An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1092–1125, 2013.

[39] Monteiro, R. D. C. and Svaiter, B. F., "Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 475–507, 2013.

[40] Monteiro, R. D., "First-and second-order methods for semidefinite programming," *Mathematical Programming*, vol. 97, no. 1-2, pp. 209–244, 2003.

[41] Nemirovski, A., "Prox-method with rate of convergence O(1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 229–251, 2004.

[42] Nesterov, Y., "A method for unconstrained convex minimization problem with the rate of convergence O (1/k2)," *Doklady AN SSSR*, vol. 269, no. 3, pp. 543–547, 1983.

[43] Nesterov, Y., *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

[44] Nesterov, Y., "Excessive gap technique in nonsmooth convex minimization," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 235–249, 2005.

[45] Nesterov, Y., "Smooth minimization of non-smooth functions," *Mathematical programming*, vol. 103, no. 1, pp. 127–152, 2005.

[46] Nesterov, Y., "Dual extrapolation and its applications to solving variational inequalities and related problems," *Math. Program.*, vol. 109, no. 2-3, Ser. B, pp. 319–344, 2007.

[47] Nesterov, Y., *Gradient methods for minimizing composite objective function*. CORE, 2007.

[48] Nesterov, Y., "Smoothing technique and its applications in semidefinite optimization," *Mathematical Programming*, vol. 110, no. 2, pp. 245–259, 2007.

[49] Niess, S., "Counting monochromatic copies of K_4: a new lower bound for the Ramsey multiplicity problem," *arXiv preprint arXiv:1207.4714*, July 2012.

[50] Nocedal, J. and Wright, S. J., *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, Springer, second ed., 2006.

[51] O'Donoghue, B. and Candès, E., "Adaptive restart for accelerated gradient schemes," *Foundations of Computational Mathematics*, pp. 1–18, 2013.

[52] Povh, J., Rendl, F., and Wiegele, A., "A boundary point method to solve semidefinite programs," *Computing*, vol. 78, pp. 277–286, 2006. 10.1007/s00607-006-0182-2.

[53] Povh, J. and Rendl, F., "Copositive and semidefinite relaxations of the quadratic assignment problem," *Discrete Optimization*, vol. 6, no. 3, pp. 231 – 241, 2009.

[54] Rockafellar, R. T., *Convex Analysis*. Princeton, NJ: Princeton University Press, 1970.

[55] Rockafellar, R. T., "On the maximal monotonicity of subdifferential mappings," *Pacific J. Math.*, vol. 33, pp. 209–216, 1970.

[56] Rockafellar, R. T., "Augmented lagrangians and applications of the proximal point algorithm in convex programming," *Mathematics of operations research*, vol. 1, no. 2, pp. 97–116, 1976.

[57] Rockafellar, R. T., "Monotone operators and the proximal point algorithm," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[58] Roh, T. and Vandenberghe, L., "Discrete transforms, semidefinite programming, and sum-of-squares representations of nonnegative polynomials," *SIAM Journal on Optimization*, vol. 16, no. 4, pp. 939–964, 2006.

[59] Solodov, M. V. and Svaiter, B. F., "A hybrid approximate extragradient–proximal point algorithm using the enlargement of a maximal monotone operator," *SetValued Analysis*, vol. 7, no. 4, pp. 323–345, 1999.

[60] Solodov, M. V. and Svaiter, B. F., "A hybrid projection-proximal point algorithm," *Journal of Convex Analysis*, vol. 6, no. 1, pp. 59–70, 1999.

[61] Solodov, M. V. and Svaiter, B. F., "An inexact hybrid generalized proximal point algorithm and some new results on the theory of Bregman functions," *Mathematics of Operations Research*, vol. 25, no. 2, pp. 214–230, 2000.

[62] Solodov, M. V. and Svaiter, B. F., "A unified framework for some inexact proximal point algorithms," *Numerical Functional Analysis and Optimization*, vol. 22, no. 7-8, pp. 1013–1035, 2001.

[63] Svaiter, B. F., "A family of enlargements of maximal monotone operators," *Set-Valued Analysis*, vol. 8, pp. 311–328, 2000. 10.1023/A:1026555124541.

[64] Toh, K. C., Todd, M., and Tütüncü, R. H., "SDPT3 - a MATLAB software package for semidefinite programming," *Optimization Methods and Software*, vol. 11, pp. 545–581, 1999.

[65] Tseng, P., "On accelerated proximal gradient methods for convex-concave optimization," *submitted to SIAM Journal on Optimization*, 2008.

[66] Wallin, R., Hansson, A., and Johansson, J. H., "A structure exploiting preprocessor for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma," *Automatic Control, IEEE Transactions on*, vol. 54, pp. 697–704, April 2009.

[67] Wen, Z., Goldfarb, D., and Yin, W., "Alternating direction augmented lagrangian methods for semidefinite programming," *Mathematical Programming Computation*, vol. 2, pp. 203–230, 2010. 10.1007/s12532-010-0017-1.

[68] Zhao, X.-Y., Sun, D., and Toh, K.-C., "A Newton-CG augmented lagrangian method for semidefinite programming," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1737–1765, 2010.